ANGULAR WITH MA
IT'S MAPTASTIC

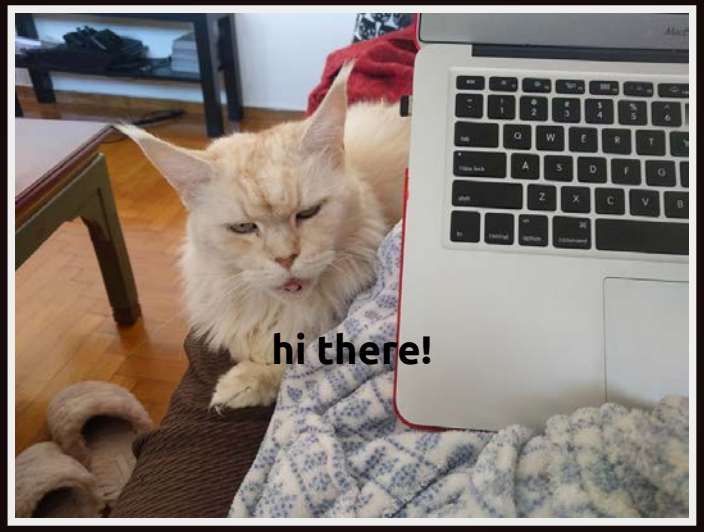
Katerina Skroumpelou

Upstream, Greece

@psybercity

a little bit about myself

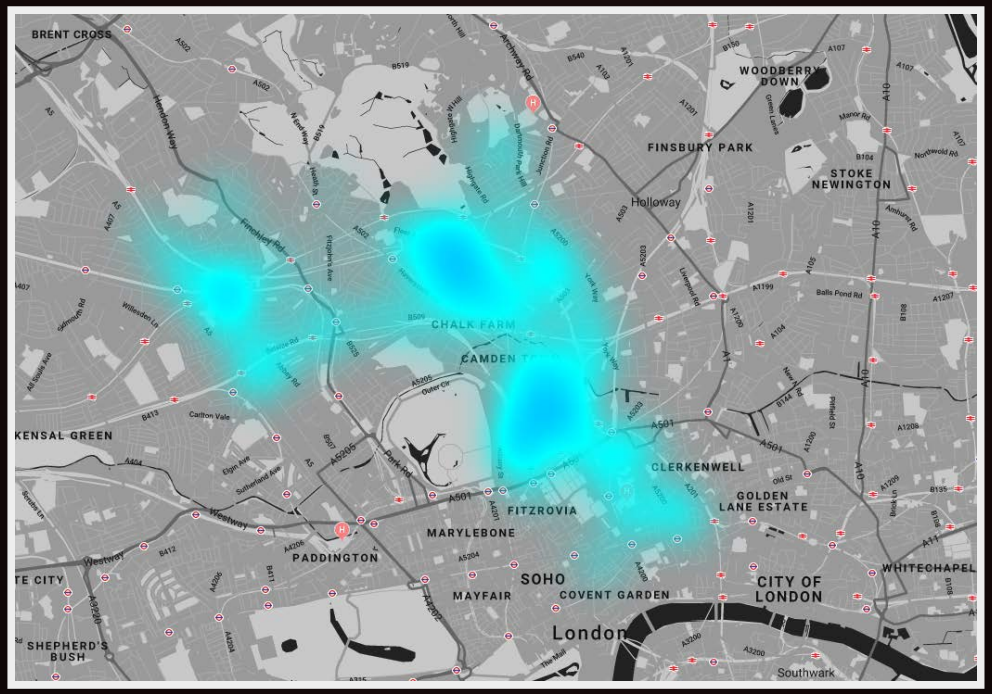
- front-end web dev at Upstream, Greece
- studied spatial analysis and spatial data viz at CASA, UCL
- worked a lot with AngularJS and maps APIs
- lives in Athens, Greece
- codes with cat on lap



summary

We are going to see how easy it is to add a map to your Angular application.

Then, we'll se how you can add some data to your map, visualized too!



summary

- y u need maps?
- online map libs & what to choose
- how to integrate
- gimme teh codez!
- some data viz & examples
- map interactions

y u need maps

- maps are everywhere (think of an app)
- maps are used to show data
- maps are nice and interactive

online map libs

some of them:

- mapbox >pricing
- leaflet >FOSS, simple to use
- OpenLayers >FOSS, complex, can handle complexity
- Google Maps JS API >pricing, can handle complexity, google APIs!
- d3.js >not really a map lib, can handle spatial data viz
- ArcGIS JS API >enterprise software, powerful
- CartoDB >not really a map lib, great for spatial data viz

today's presentation

we are going to work with the Google Maps JS API

Angular integration

asynchronous script loading?

synchronous script loading?

dynamic asynchronous script loading?

script loading

The maps APIs are loaded with a script.

The script defines a global variable/object, that is the map and all its functions (google.map)

There are two ways to load a script, synchronously and asynchronously:

asynchronously means that your page will continue doing what a page has to do, while the script is being loaded

synchronously means that your page will stop everything and wait for the script to be loaded in order to continue

script loading

Once your script is **loaded** and the global object is **de ned**, normally you could be able to access it from anywhere in your js code, like this

```
let map = new google.maps.Map( ..details here.. );
```

script loading

There are two ways, thus, to know when your script is loaded.

1_sync

You can load the script **synchronously**.

- rest of your code waits for the script to be loaded
- when it's executed the global object is defined.

```
<script src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY"></script>
```

This is not advisable, since it can slow down your app, and in the case the script is never loaded, well, then a great bit of your app is never loaded.

script loading

2_async

You can load the script **asynchronously**.

In this case, you have to add a callback function, as shown in the example from the Google Maps JS API docs:

```
<script src="https://maps.googleapis.com/maps/api/js?key=API_KEY&callback=initMap" async="" defer="">
</script>
```

When the script is loaded, it will search a global function, named here `initMap`, to execute it. In the scope of this function, the `google.maps` object will be available.

script loading

2_async

However, accessing a globally available function from the script loader is not so straightforward in Angular.

Also, since we are usually targeting performance, we prefer not to load things until they are actually needed.

Which leads us to...

dynamic asynchronous **script loading**!

dynamic script loading

How to in Angular?

We create a service that loads the script!

```
public loadScript(url, id, c): void {
   if (!document.getElementById(id)) {
     const script = document.createElement('script');
     script.type = 'text/javascript';
     script.src = url;
     script.id = id;
     script.addEventListener('load', function (e) {
        c(null, e);
     }, false);
     document.head.appendChild(script);
   }
 }
```

dynamic script loading

`script.addEventListener('load', callback)` fires on script load and calls the callback function that we pass to it!

```
[ScriptLoadService object].loadScript(mapsApiUrl, 'map-script-id', () => {
    const maps = window['google']['maps'];
    this.map = new maps.Map( ... details here ...);
}
```

Notice how instead of referencing directly the `google.maps` object (as we could and would be allowed to), we are reading it explicitly from the window object. If we reference it directly, the linter will complain because it cannot see its definition.

step by step

1. create a script loading service with the script loading function
   `@Injectable() export class ScriptLoadService {}`
2. import it as a provider to your app module
   `providers: [ScriptLoadService]`
3. define it in your constructor in the component that uses the map
   `constructor(private load: ScriptLoadService) {}`

step by step

4. create a div where you will put the map and add a template reference variable to it

```
< div class="map" #mapElement>< / div >
```

5. give height to the div (empty divs usually have 0 height)

```
.map { height: 90vh; }
```

6. select your div in the component

```
@ViewChild('mapElement') mapElm: ElementRef;
```

7. place your map on the div

```
this.map = new maps.Map(this.mapElm.nativeElement, {.. details here ..});
```

the full component code

```
import { Component, AfterViewInit, ViewChild, ElementRef } from '@angular/core';
import { ScriptLoadService } from '../script-load.service';

const your_API_key = 'xxx';
const url = 'https://maps.googleapis.com/maps/api/js?key=' + your_API_key;

@Component({
  selector: 'app-g-map',
  templateUrl: './g-map.component.html',
  styleUrls: ['./g-map.component.css']
})
export class GMapComponent implements AfterViewInit  {

  @ViewChild('mapElement') mapElm: ElementRef;

  private map: any;
```

show me the map already

ok!

AfterViewInit

So, why are we using `AfterViewInit` interface, and the ViewChild decorator?

We need access to the component DOM view and its children.

from the angular docs about lifecycle hooks:
`ngAfterViewInit()` *is called once after Angular initializes the component's views and child views / the view that a directive is in.*

we use `@ViewChild('mapElement') mapElm: ElementRef;`
to let angular know that we are looking for a native DOM element.

then we hook it to `ngAfterViewInit()` where the view children have been created.

spatial data

A map is -essentially- a two dimensional representation of a piece of the earth. Enriched with two dimensional representations of information about this piece of the earth. Information about roads, rivers, shapes of the earth, building blocks, etc. To scale or not.

spatial data representation formats

- Geojson
- json
- csv
- shapefile
- kml
- gml
- & more

where to nd spatial data

data.gov.uk, eurostat, geodata.gov.gr and many more.

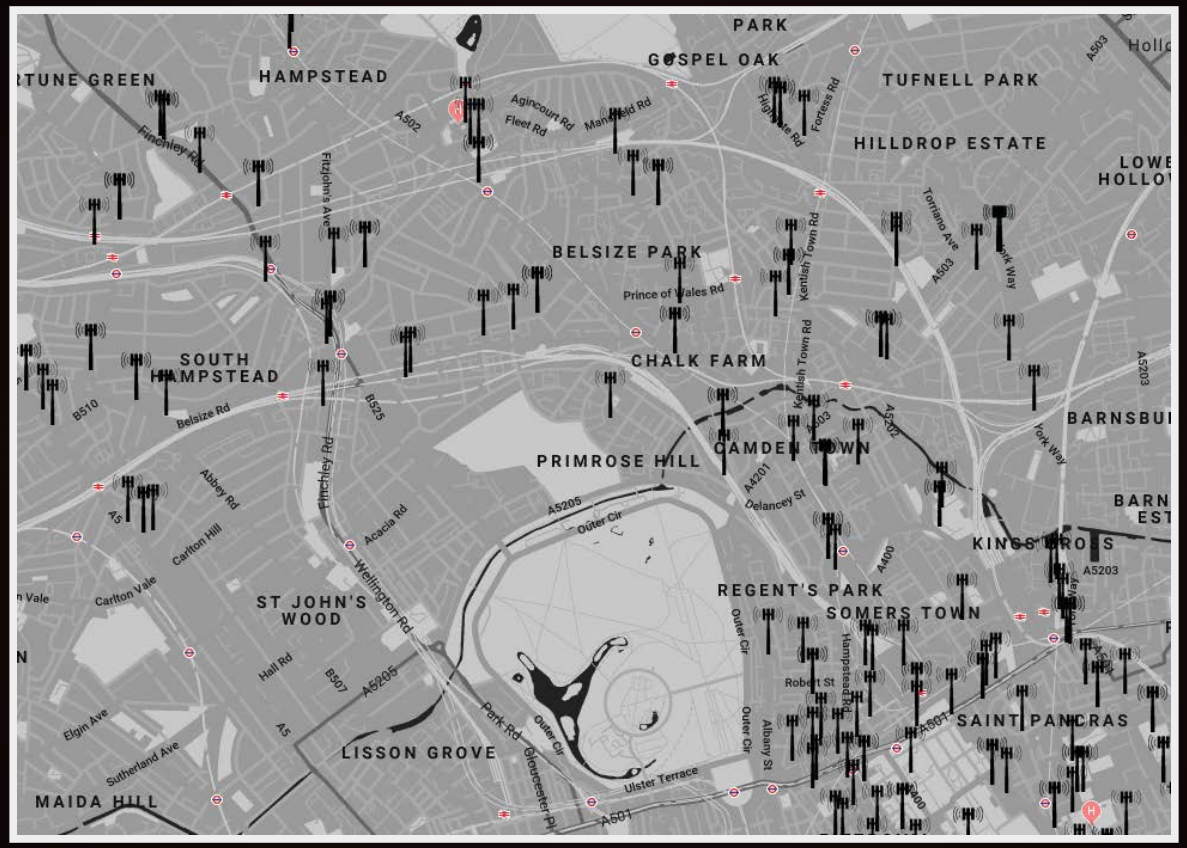
here you can find a nice list of spatial data

types of data and representations

Points, polygons, points with values, clusters (heatmaps) etc.

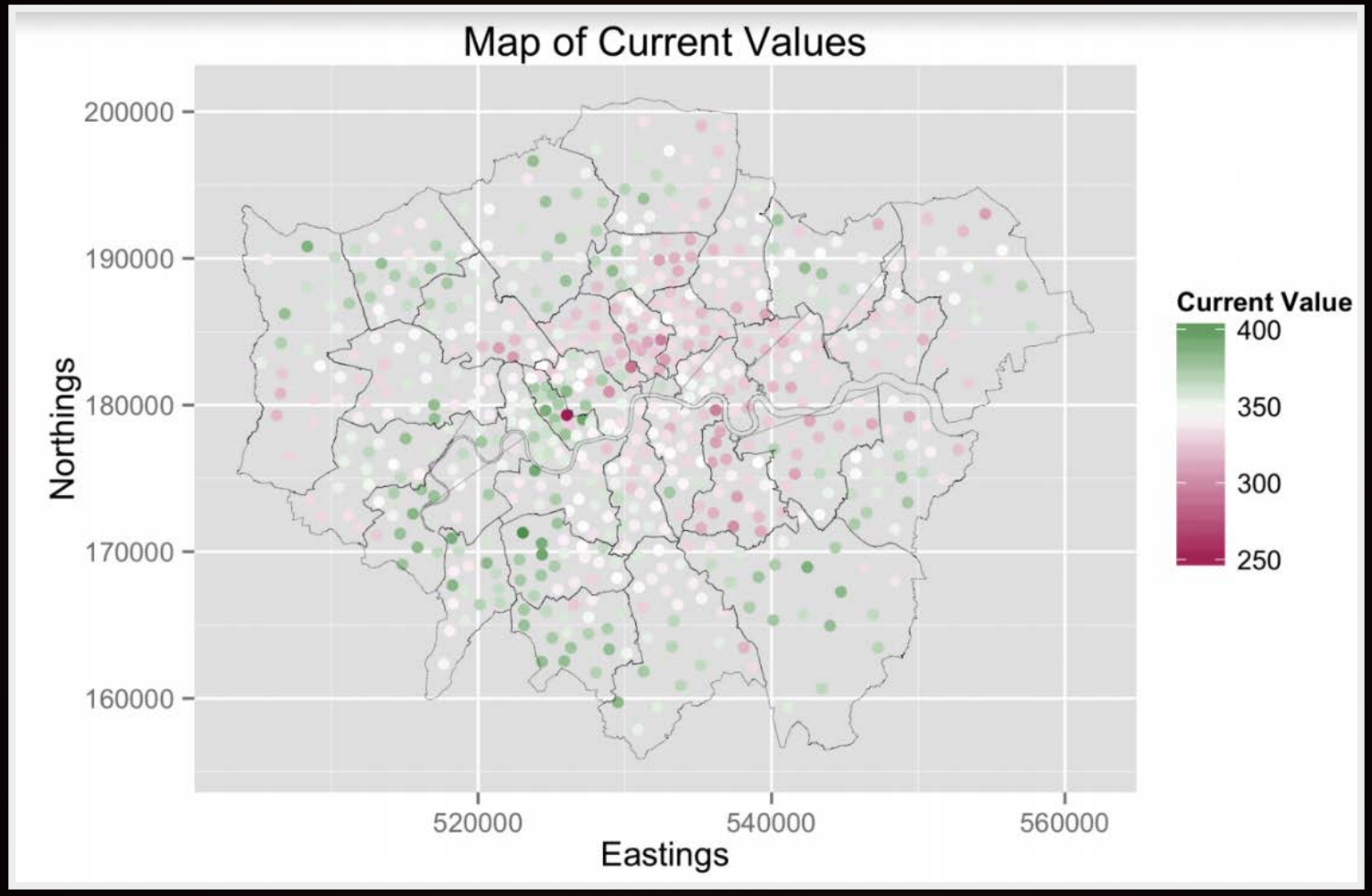
According to the format of your data (or the story your data want to tell), you can create corresponding visuals. here are some techniques.

points



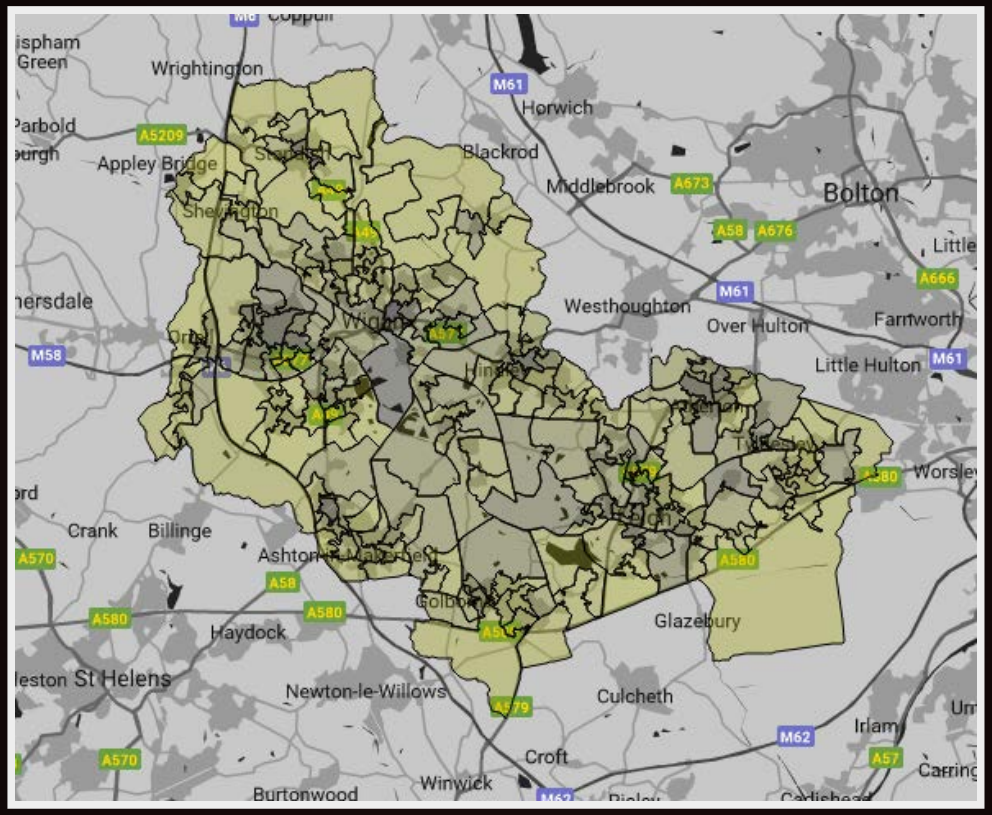
points => mobile phone masts

points / types / values



values => color

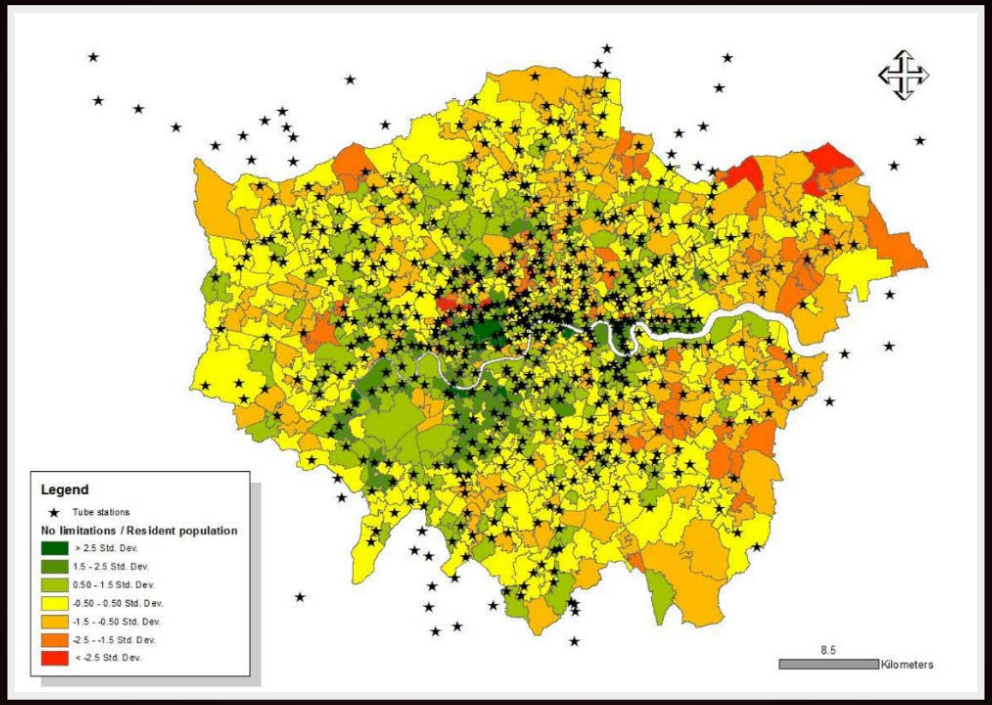
polygons



polygons => Lower layer Super Output Areas (geographic areas) in Manchester

color => loneliness and social isolation prevalence factor

polygons

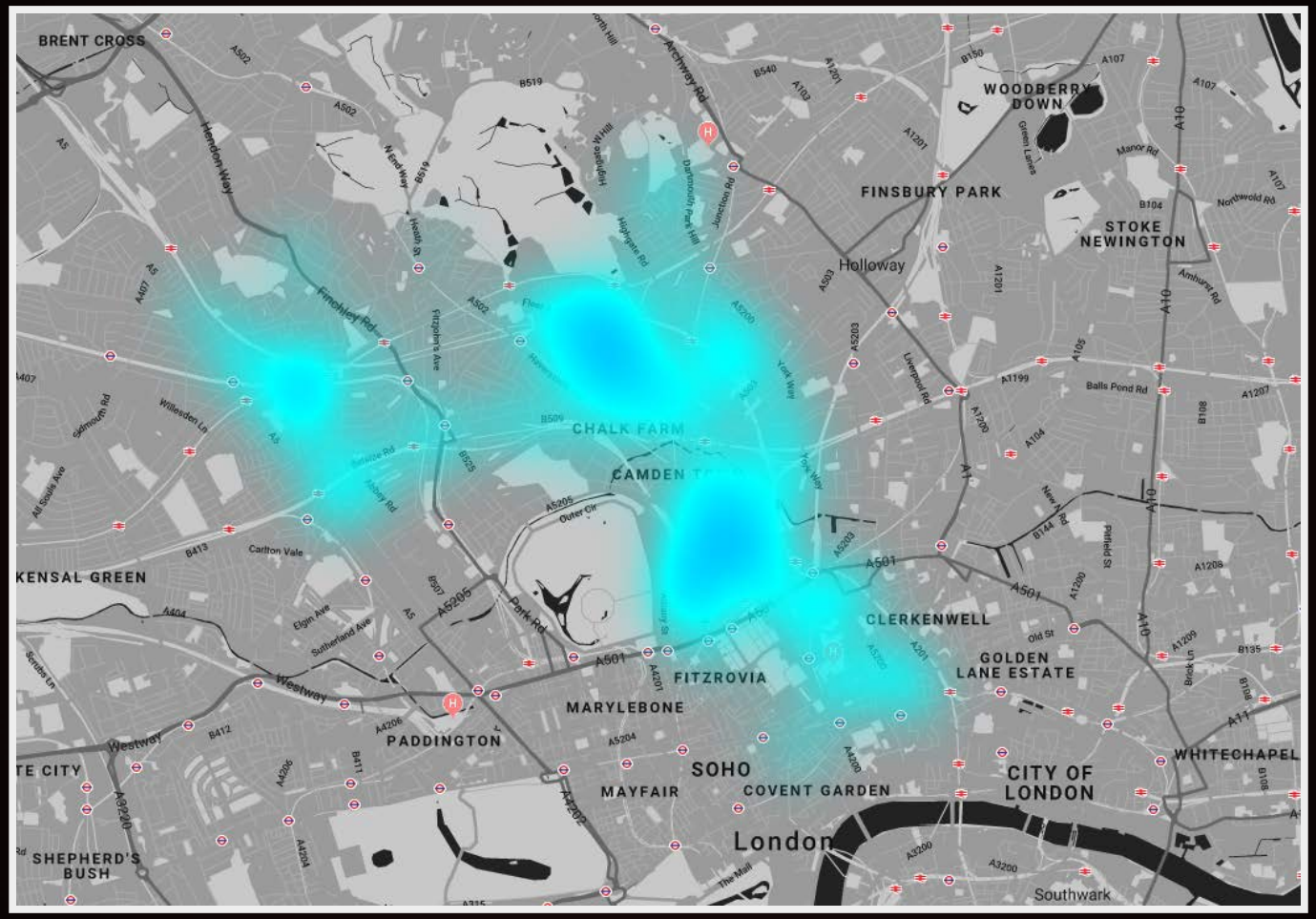


polygons => wards

color => amount of residents with no mobility limitations

points => tube stations

heatmap



letting bids

let's see how to do this

polygons - geojson


```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "OBJECTID": 112,
        "LOWERSOA": "E01006279",
        "HOUSEHOLDS": 508,
        "POPULATION": 1381,
        "AREA_NAME": "Hindley Green/Leigh Road",
        "TOWN": "Hindley Green",
        "TOWNSHIP": "Hindley Abram",
        "WARD": "Hindley Green",
        "COMMUNITY": "Hindley / Hindley Green",
        "PREVALENCE": 1.32742523,
```

polygons - geojson

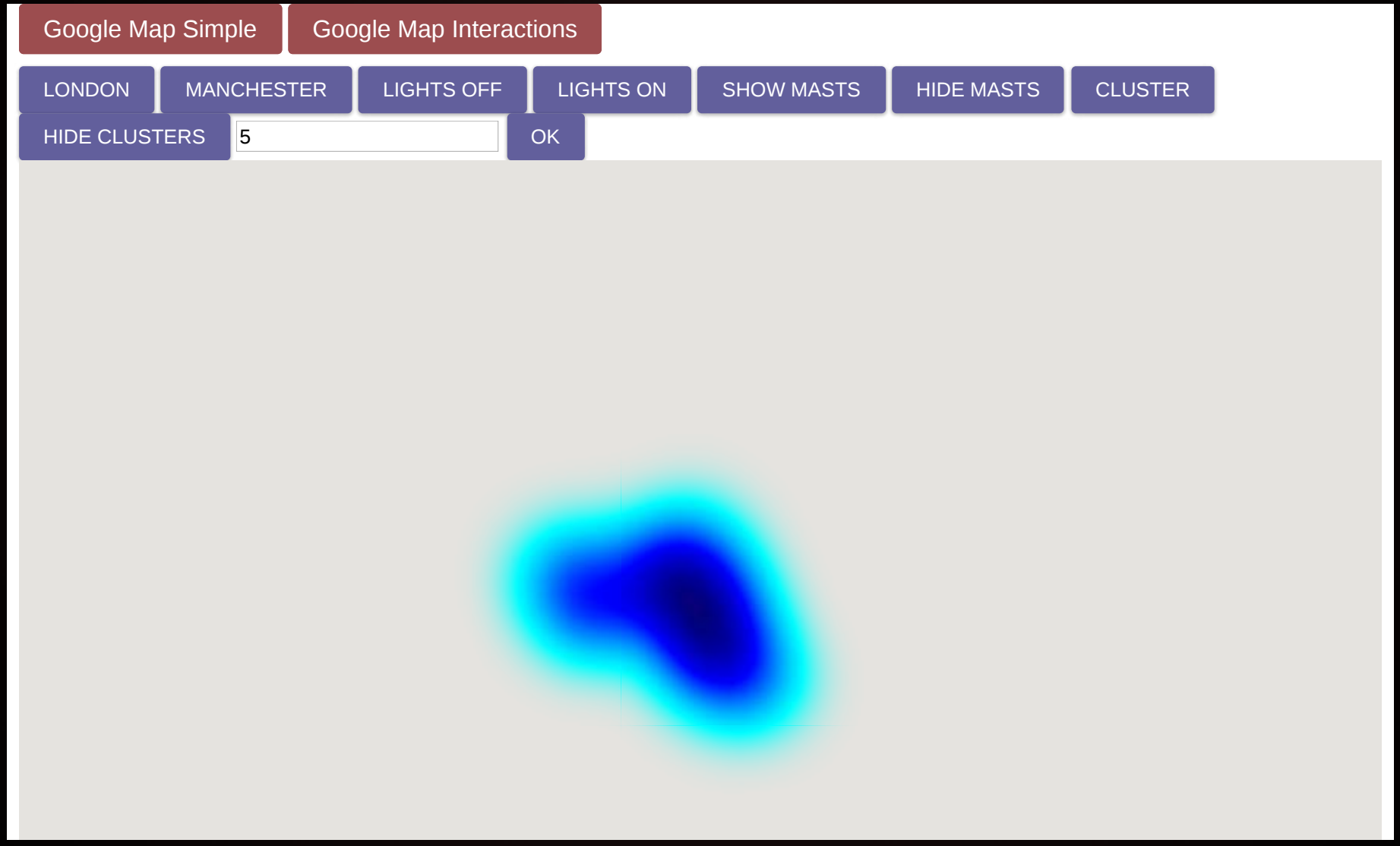
```
this.map.data.loadGeoJson('path-to-file/filename.geojson');
this.map.data.setStyle(function(feature) {

    const lon = feature.getProperty('PREVALENCE');

    const value = 255 - Math.round(mapNumber(lon, 0, 5, 0, 255));
    const color = 'rgb(' + value + ',' + value + ',' + 0 + ')';

    return {
        fillColor: color,
        strokeWeight: 1
    };
});
```

result


Google Map Simple
Google Map Interactions
LONDON
MANCHESTER
LIGHTS OFF
LIGHTS ON
SHOW MASTS
HIDE MASTS
CLUSTER
HIDE CLUSTERS
5
OK

points - shapes


```
[
  204,
  "55FE58A2-E7E9-4E92-80C3-ED2DD94901FC",
  204,
  1447082076,
  "1113",
  1447082076,
  "1113",
  null,
  "CTIL",
  "141661",
  "Thistle Hotel",
  "Bloomsbury Way",
  "WC1A 2SD",
  "E05000138",
  "Holborn and Covent Garden",
```

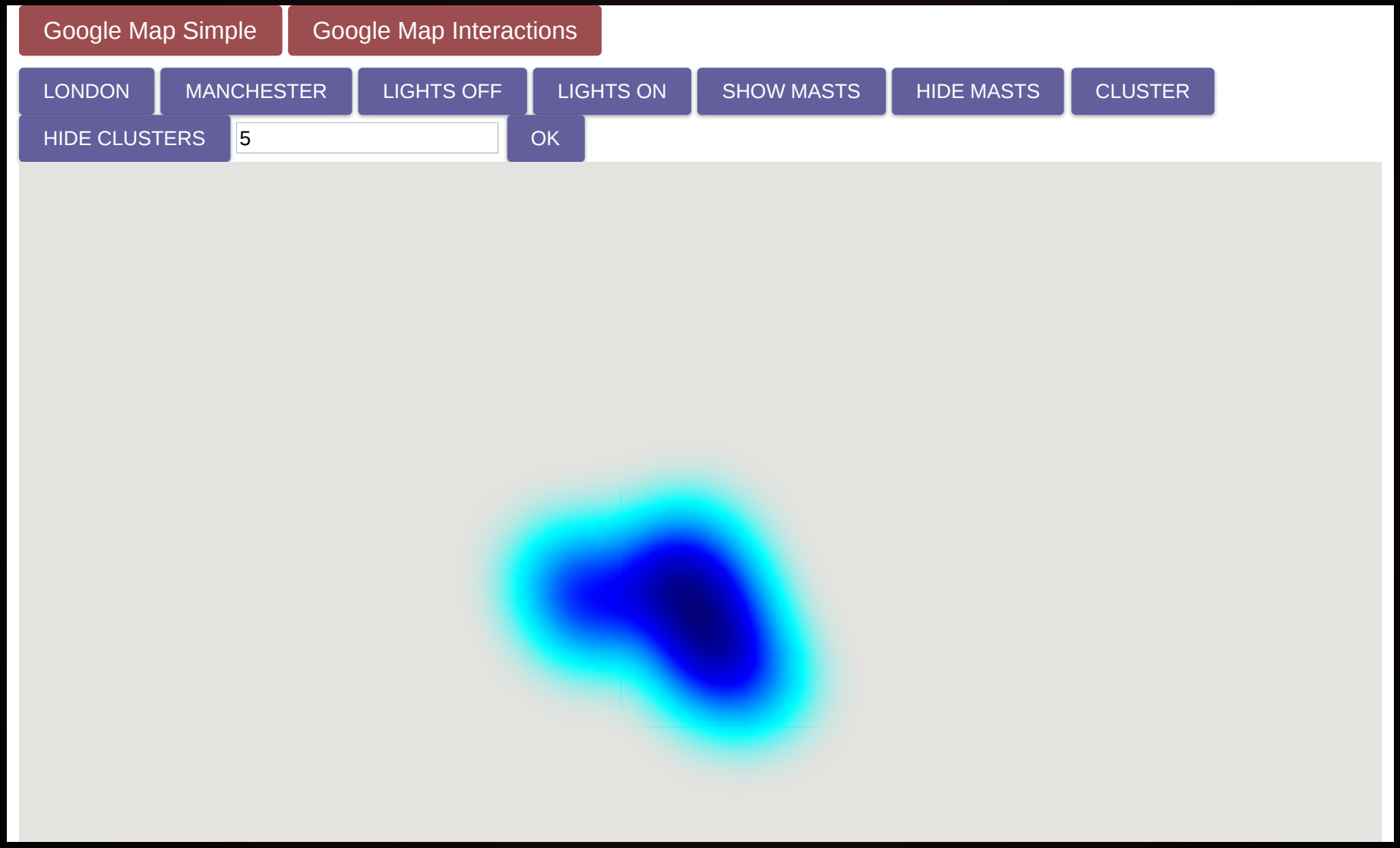
points - shapes

```
const antenna = new maps.MarkerImage('path-to-image/marker-image.png');

this.http.get('path-to-file/file.json').subscribe(data => {
    this.masts = data['data'];

    this.masts.map(x => {
        new maps.Marker({
        position: new maps.LatLng(x[18], x[17]),
        icon: antenna,
        size: new maps.Size(30, 30),
        map: this.map
        });
    });
```

result


Google Map Simple
Google Map Interactions
LONDON
MANCHESTER
LIGHTS OFF
LIGHTS ON
SHOW MASTS
HIDE MASTS
CLUSTER
HIDE CLUSTERS
5
OK

points - values - heatmap

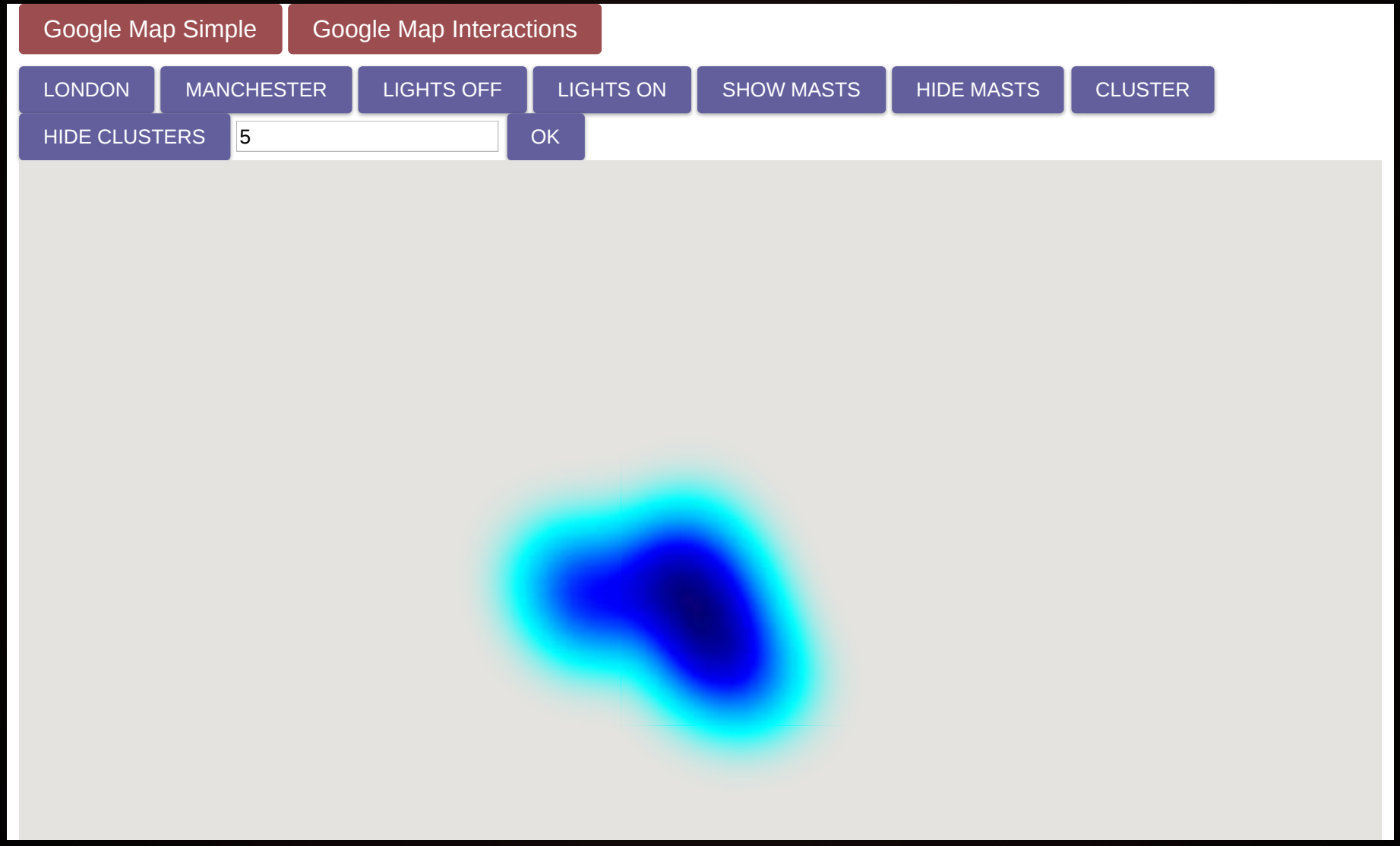
csv

```
Advert Reference,Property Reference,Property Address,Bid Opening Date,Bid Closing Date,Number
102401,19575,"Flat 181,Mayford,Oakley Square,London,NW1 1PA",23/06/2016,27/06/2016,2,4,493,60
199144,21832,"Flat 22, Oak House, Maitland Park Villas, London, NW3 2ED",15/09/2016,19/09/201
288670,29672,"Flat 13, Tapestry Apartments, 3 Canal Reach, London, N1C 4BA",29/09/2016,03/10/
```

points - values - heatmap

```
this.http.get('assets/letting.json').subscribe(data => {
  this.lettings = data['data'];
  const heatmapData = [];
  this.lettings.map(x => {
    heatmapData.push({
      location: new maps.LatLng(x[24], x[23]),
      weight: parseInt(x[15], 10)
    });
  });
  const heatmap = new maps.visualization.HeatmapLayer({
    data: heatmapData
  });
  heatmap.set('gradient', customGradient);
  heatmap.set('radius', 70);
  heatmap.set('opacity', 1);
  heatmap.setMap(this.map);
```

result


Google Map Simple
Google Map Interactions
LONDON
MANCHESTER
LIGHTS OFF
LIGHTS ON
SHOW MASTS
HIDE MASTS
CLUSTER
HIDE CLUSTERS
5
OK

point clustering

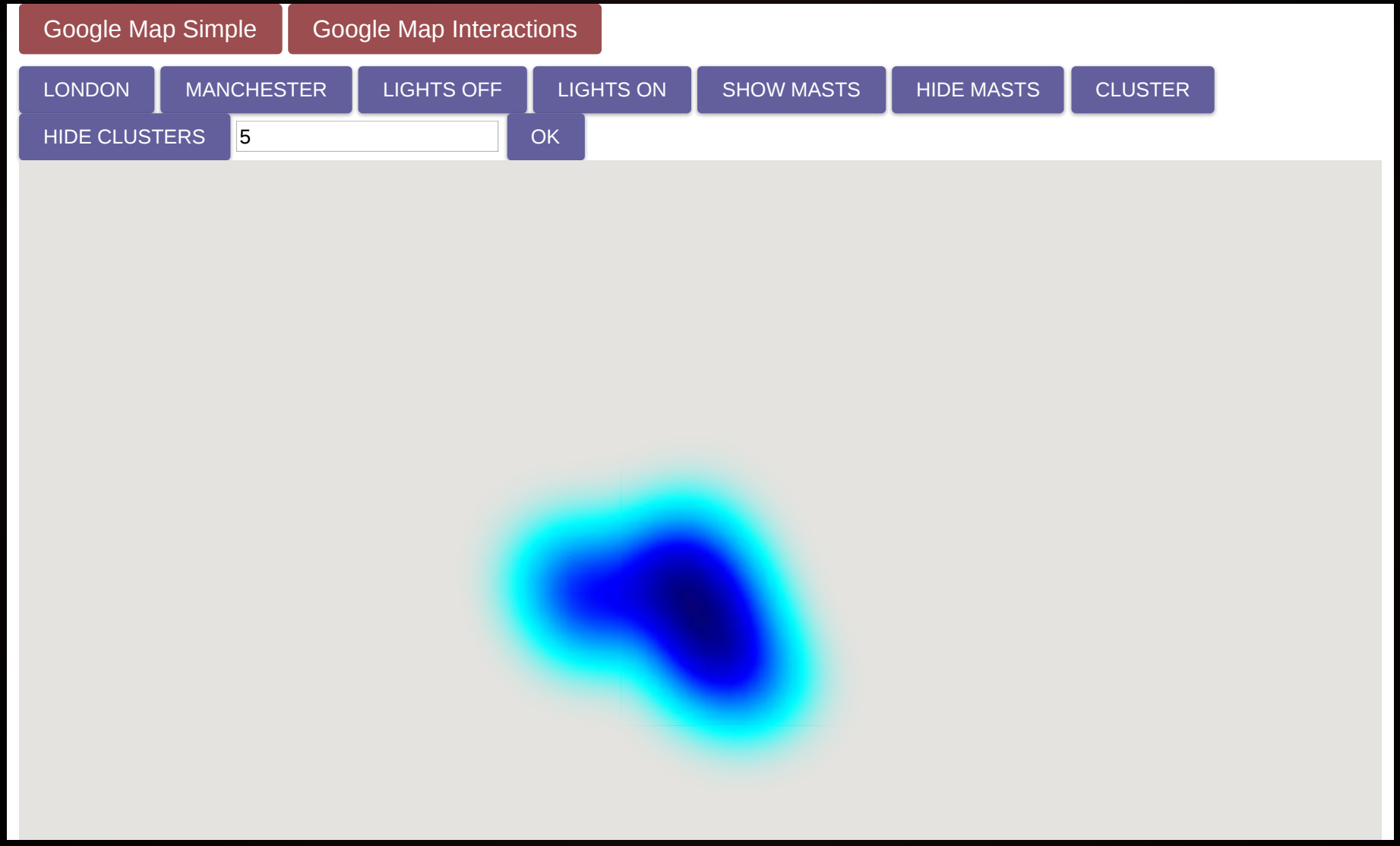
if your points are too many, you can display them in clusters, using the MarkerClusterer library.

>>*divides the map into a grid of a certain size and groups the markers into each square,* **based on the distance between them**

you may specify the grid size to create larger or smaller clusters according to what you need to visualize

```
const MarkerClusterer = window["MarkerClusterer"];
let markerCluster = new MarkerClusterer(this.map, this.markers, {imagePath: 'assets/m'});
markerCluster.setGridSize(clust);
```

result


Google Map Simple
Google Map Interactions
LONDON
MANCHESTER
LIGHTS OFF
LIGHTS ON
SHOW MASTS
HIDE MASTS
CLUSTER
HIDE CLUSTERS
5
OK

other methods

1. change map colors

```
const darkmap = new maps.StyledMapType(styledMap, {name: 'Dark Map'});
this.map.mapTypes.set('dark_map', darkmap);
this.map.setMapTypeId('dark_map');
```

2. set focus buttons

```
<button (click)="focus()">London</button>
focus() {
   this.map.setCenter(this.coords(51.5616, -0.14));
}
```

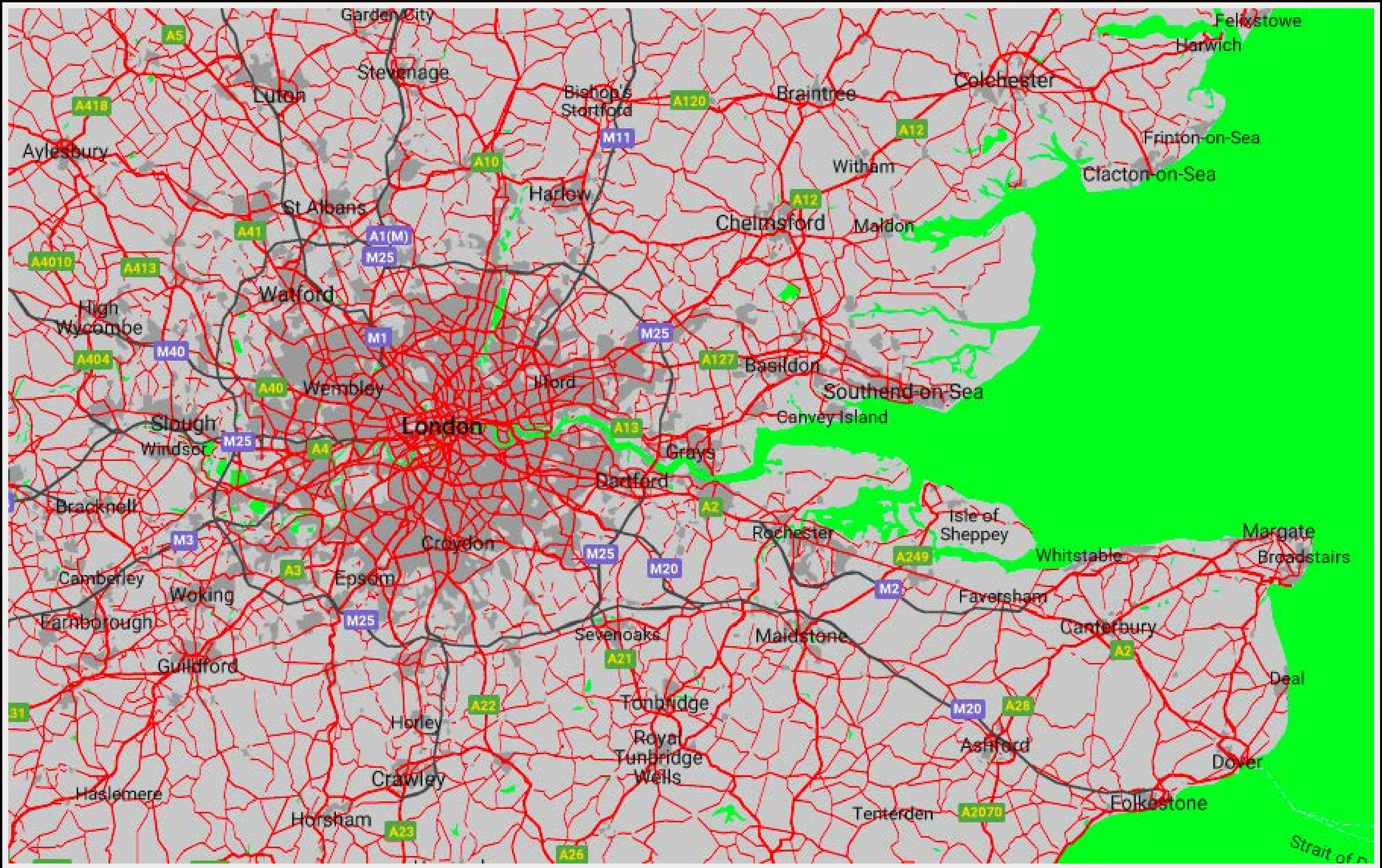
color change

note how you can choose the map features you want and color them accordingly.

you can hide and show features such as points of interest, businesses, color the sea green and the roads red.

```
{
  featureType: 'water',
  elementType: 'geometry.fill',
  stylers: [{color: '#00ff00'}]
},
{
  featureType: 'road.highway',
  elementType: 'geometry',
  stylers: [{color: '#ff0000'}]
},
```

result



Garden City
Stevenage
Luton
Bishop's Stortford
Braintree
Colchester
Felixstowe
Harwich
Aylesbury
Harlow
Witham
Frinton-on-Sea
Clacton-on-Sea
St Albans
Chelmsford
Maldon
Watford
High Wycombe
Basildon
Southend-on-Sea
Canvey Island
Wembley
Ilford
London
Slough
Windsor
Grays
Dartford
Bracknell
Croydon
Rochester
Isle of Sheppey
Whitstable
Margate
Broadstairs
Camberley
Woking
Epsom
Faversham
Farnborough
Sevenoaks
Maidstone
Canterbury
Guildford
Deal
Tonbridge
Horley
Royal Tunbridge Wells
Ashford
Dover
Crawley
Haslemere
Horsham
Tenterden
Folkestone
A5
A418
A41
A4010
A413
A404
M40
A40
M25
A4
M3
A3
A10
A1(M)
M1
A120
M11
A12
A127
A13
A2
M20
A249
M2
A21
A22
A28
A2070
A23
A26

drawing

last but not least, you can **draw** on your map!

you can create your **custom** drawing tools

and you can **save** your creations!

drawing

all you need is the drawing library

at the end of your google maps URL, append `'&libraries=drawing'`

initialize the DrawingManager, and add it to your map!

```
let drawingManager = new maps.drawing.DrawingManager({
  drawingMode: null,
  drawingControl: false
});
drawingManager.setMap(this.map);
```

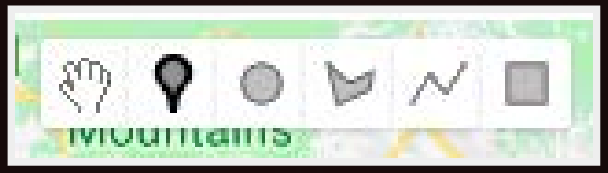
`drawingControl: false` because we'll be creating our own custom toolset!

toolsets - default

the API offers the following types of drawing modes: marker, circle, polygon, polyline, rectangle

which can be specified by, eg.

```
drawingManager.setDrawingMode(maps.drawing.OverlayType.MARKER)
```



toolsets - default

using these default modes, you can create your own custom tools!

```
<div id="draw-buttons">
  <img src="assets/point.png" (click)="draw('marker')">
  <img src="assets/polygon.png" (click)="draw('polygon')">
  <img src="assets/square.png" (click)="draw('square')">
  <img src="assets/line.png" (click)="draw('polyline')">
  <img src="assets/circle.png" (click)="draw('circle')">
  <img src="assets/cat.png" (click)="draw('cat')">
  <img src="assets/pan.png" (click)="draw('pan')">
  <img src="assets/save.png" (click)="draw('save')">
</div>
```

toolsets - default

and create your customized shapes (different coloring, different attributes, different markers)

```
case 'cat':
  this.drawingManager.setDrawingMode(maps.drawing.OverlayType.MARKER);
  let cat = new maps.MarkerImage('assets/cat.png');
  this.drawingManager.setOptions({
    markerOptions: {
      icon: cat,
      clickable: true,
      draggable: true
    }
  });
  break;
case 'polygon':
  this.drawingManager.setDrawingMode(maps.drawing.OverlayType.POLYGON);
  this.drawingManager.setOptions({
    polygonOptions: {
      fillColor: '#9c4d4f',
```

saving our creation

the customization can be proven helpful when we want to save our creations

we will be saving our creations in a universal geodata format, **geoJSON**

geoJSON allows us to specify properties for our features (shapes)

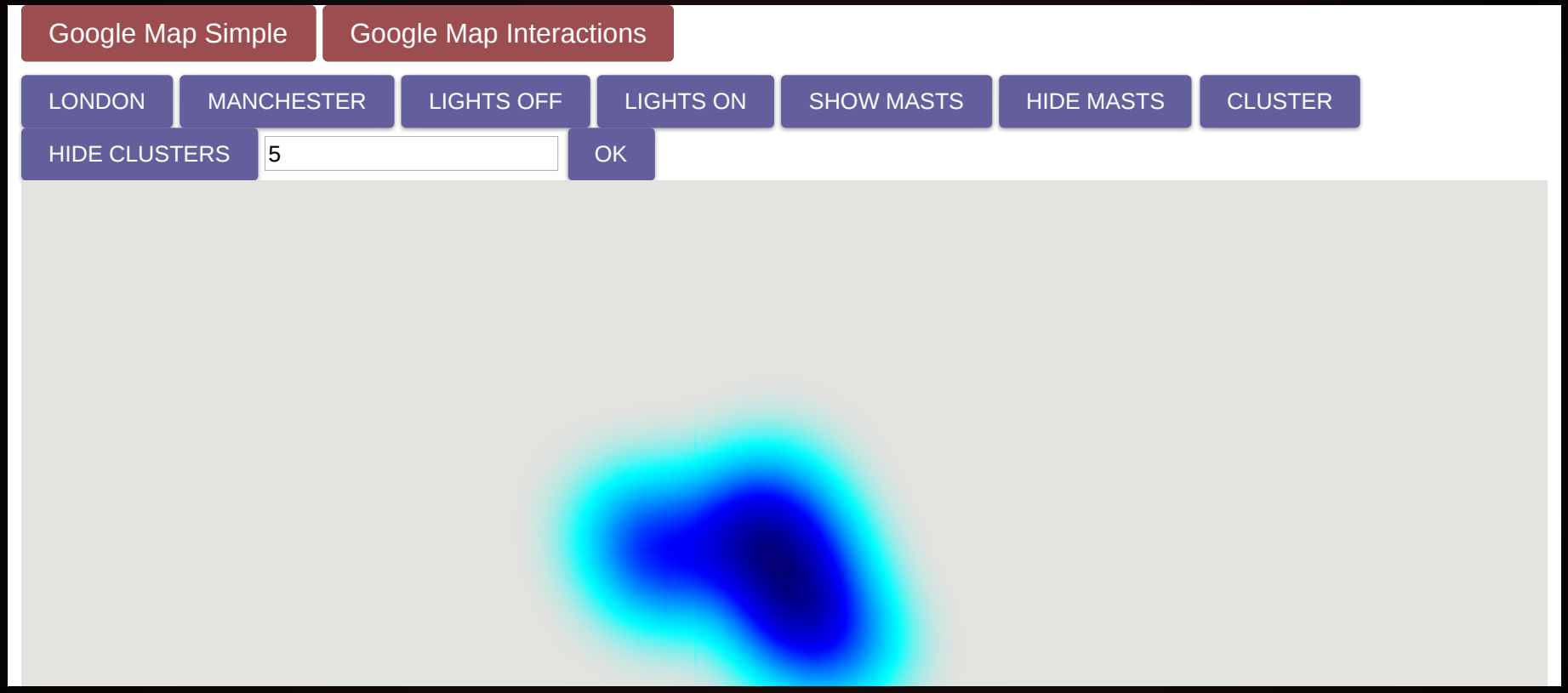
saving our creation

```
map.data.add(new maps.Data.Feature({
  geometry: new maps.Data.Polygon([points]),
  properties: {
    color: '#ff00ff',
    type: 'playground'
  }
}));

...

map.data.toGeoJson(function (obj) {
  postToServer(obj);
});
```

the map


Google Map Simple
Google Map Interactions
LONDON
MANCHESTER
LIGHTS OFF
LIGHTS ON
SHOW MASTS
HIDE MASTS
CLUSTER
HIDE CLUSTERS
5
OK

what we learned about

- about map libs
- how to load map
- how to visualize data
- how to interact
- how to draw and save drawing

-the end-

Katerina Skroumpelou

Upstream, Greece

@psybercity

github.com/mandarini

Slides:
https://mandarini.github.io/iJSPres

Code from presentation:
https://github.com/mandarini/map

credit - sources

- ViewChildren & ContentChildren by @jawache
- Loneliness in LSOA
- Letting bids
- Mobile phone masts in camden
- Google Maps JS API docs
- John Snow and the Soho cholera map
- xkcd - Heatmap