AGM MODULE

- Maps Javascript API

The Maps JavaScript API lets you customize maps with your own content and imagery for display on web pages and mobile devices. The Maps JavaScript API features four basic map types (roadmap, satellite, hybrid, and terrain) which you can modify using layers and styles, controls and events, and various services and libraries.
For more Information:
https://developers.google.com/maps/documentation/javascript/tutorial

How to start:

1. Install Agm Module in angular using `` `npm install @agm/core` ``.
2. Add `import {AgmCoreModule} from '@agm/core';` in `app.module.ts`
3. Add

```
  AgmCoreModule.forRoot({
    apiKey: 'API_KEY'
  })
```

   in `imports array` of `app.module.ts`

4. Add a service map.service.ts to load the maps javascript API

```
import { Injectable } from '@angular/core';
import { Jsonp, Headers } from '@angular/http';
import { Subject, Observable ,  throwError } from 'rxjs';

@Injectable()
export class MapService {
 private googleReadyObservable;
 constructor(
   private jsonp: Jsonp,
   private http: HttpClient
 ) {
   this.googleReadyObservable = new Subject();
   this.jsonp

.get(`https://maps.googleapis.com/maps/api/js?key=API_KEY&callback=JSONP_CALLBA
CK`).pipe(
     retry())
     .subscribe(res => {
       if (res.status === 200) {
         this.googleReadyObservable.complete();
       }
     });
 };

 googleReady() {
   return this.googleReadyObservable;
 };
```

5. Add a new Component to hold the map eg: map.component.ts

In map.component.ts

```
import { Component, OnInit } from '@angular/core';
import { MapService } from './map.service'

declare const google: any;

@Component({
 selector: 'app-map',
 templateUrl: './map.component.html',
 styleUrls: ['./map.component.css']
})
export class MapComponent implements OnInit {
 map: any;

 constructor(private mapService: MapService) { }

 ngOnInit() {
   this.initMap()
 }
 ngOnChanges() {
   this.initMap()
 }
  initMap() {
   this.mapService.googleReady()
       .subscribe(
           null,
           err => console.log(err),
           () => {

               var myLatlng = new google.maps.LatLng(-34.397, 150.644);
               const mapOptions = {
                   zoom: 8,
                   center: myLatlng,
                   scrollwheel: false
               };
               try {
                   this.map = new
google.maps.Map(document.getElementById('map'), mapOptions);
               }
               catch (E) {
                   this.map = null;
                   console.log('Reload')
               }


           }
       );
 }
```

```
}
```

In map.component.html

```html
<div class="container">
   <div id="map">
   </div>
</div>
```

In map.component.css

```css
  #map {
   height: 500px;
 }
 html, body {
   height: 100%;
   margin: 0;
   padding: 0;
 }
```

Ensure that you have declared a `div` element in which the map will appear on the screen. Make sure that the `div` element for the map has a height. By default, `div` elements are created with a height of 0, and are therefore invisible.

1. Adding a marker in map

```javascript
var myLatlng = new google.maps.LatLng(-34.397, 150.644);

let markerElement = new google.maps.Marker({
                  position: myLatlng,
                  map: this.map,
                title : 'Sample Marker',
            })
            markerElement.setMap(this.map);
```

This will place a marker at position specified in `mylatlng`.

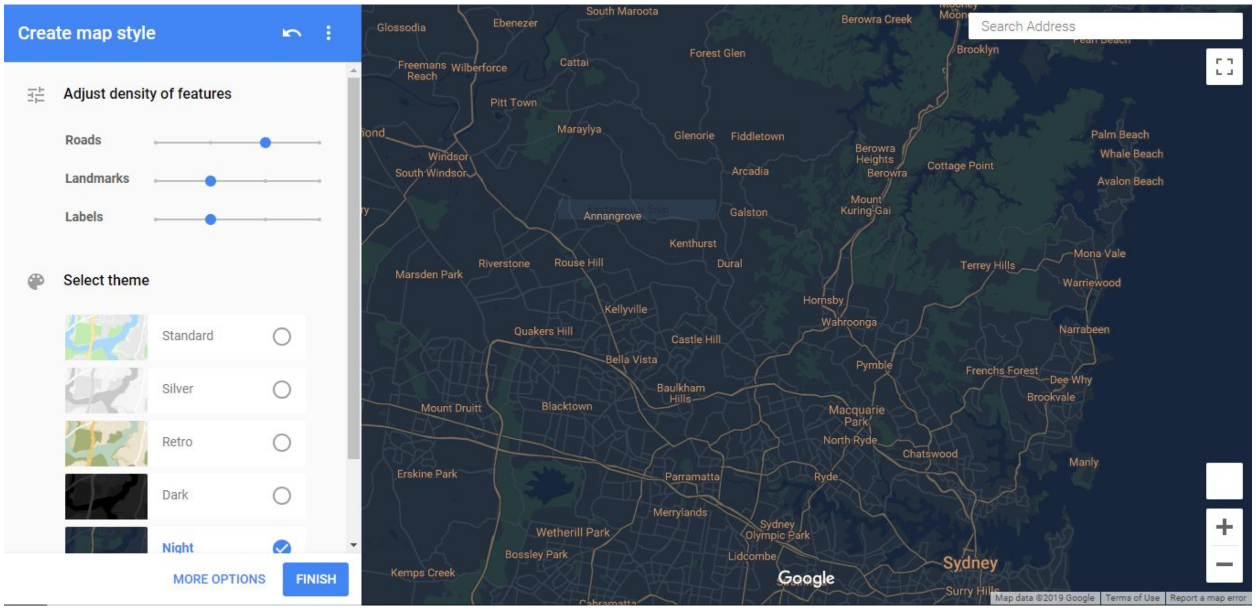
2. **Adding an infowindow to the marker element**

```
 let infowindow = new google.maps.InfoWindow({
      content:'<div id="content">Sample</div>'
     })
     markerElement.setMap(this.map);
     markerElement.addListener('mouseover', function () {
     infowindow.open(this.map, markerElement);
});
```

3. **Stylizing the map**

   With style options you can customize the presentation of the standard Google map styles, changing the visual display of features like roads, parks, businesses, and other points of interest. This means that you can emphasize particular components of the map.

   We can use the Maps Platform Styling Wizard as a quick way to generate a JSON styling object.



After choosing required styles we can import the JSON from this wizard and use it inside the styles for our map

```
const mapOptions = {
                        zoom: this.zoom,
                        center: myLatlng,
                        scrollwheel: false,
                        styles :[
                           “Add your custom styles here”
                          ]
                  };
 this.map = new google.maps.Map(document.getElementById('map'), mapOptions);
```

4. Polyline to draw a line on the map

A Polyline is a series of connected line segments. Polylines are useful to represent routes, paths, or other connections between locations on the map.

```
var location = [

          {lat: 37.772, lng: -122.214},

          {lat: 21.291, lng: -157.821},
          {lat: -18.142, lng: 178.431},

          {lat: -27.467, lng: 153.027}

        ];

var flightPath = new google.maps.Polyline({
                                           path: location,
                                           geodesic: true,
                                           strokeColor: '#333333',
                                           strokeOpacity: 1.0,
                                           strokeWeight: 2
                                       });

flightPath.setMap(this.map);
```

5. **Add Polygon and shapes to map**

```
           //To add simple polygon in map

           //Add Coordinates of polygon in an array
           var triangleCoords = [
             {lat: 25.774, lng: -80.190},
             {lat: 18.466, lng: -66.118},
             {lat: 32.321, lng: -64.757}
           ];
           var triangle = new google.maps.Polygon({
             paths: triangleCoords,
             strokeColor: '#FF0000',
             strokeOpacity: 0.8,
             strokeWeight: 2,
             fillColor: '#FF0000',
             fillOpacity: 0.35,
             editable: true // Polygon can be edited, set to false if no
editing is required.
           });
           triangle.setMap(this.map);
```

Ref:
https://developers.google.com/android/reference/com/google/android/gms/maps/model/PolygonOptions

6. Add Drawing Library

The DrawingManager class provides a graphical interface for users to draw polygons, rectangles, polylines, circles, and markers on the map.
The Drawing Tools are a self-contained library, separate from the main Maps API JavaScript code. To use the functionality contained within this library, you must first load it using the libraries parameter in the Maps API bootstrap URL.

```
<script type="text/javascript"

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

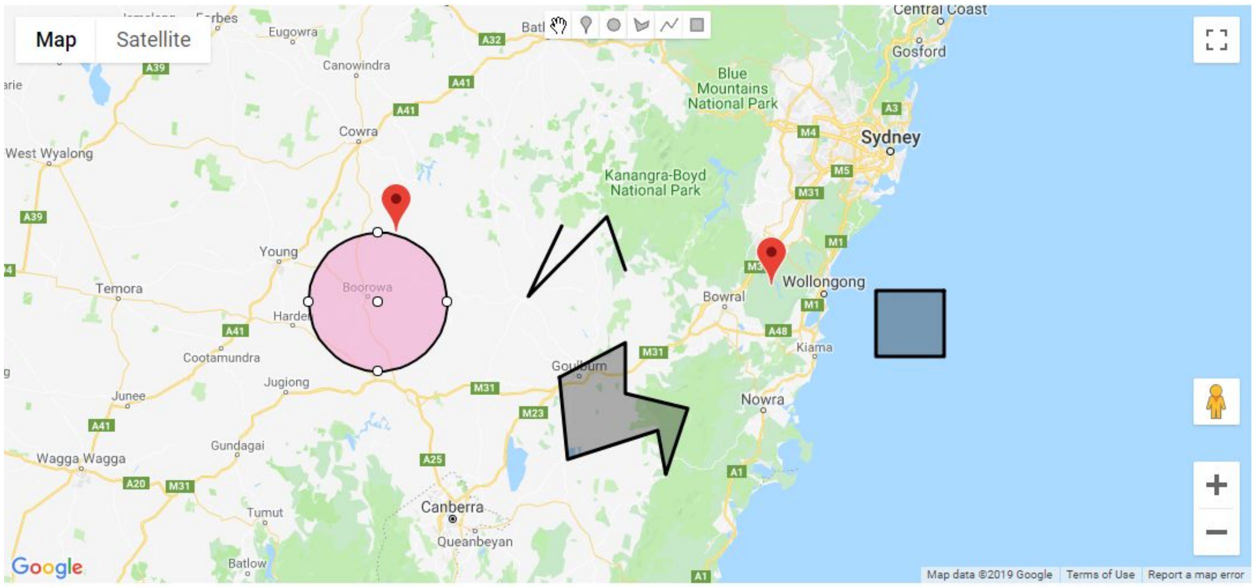
After adding libraries parameter to URL we can use Drawing manager as follows:

```
var drawingManager = new google.maps.drawing.DrawingManager();
drawingManager.setMap(map);
```

Example:

```
var drawingManager = new google.maps.drawing.DrawingManager({
          drawingMode: google.maps.drawing.OverlayType.MARKER,
          drawingControl: true,
          drawingControlOptions: {
            position: google.maps.ControlPosition.TOP_CENTER,
            drawingModes: ['marker', 'circle', 'polygon', 'polyline',
'rectangle'] // This will create a drawing manager with options for drawing
marker, Circle, Polygon, Polyline and Rectangle.
           },
           circleOptions: {
             fillColor: '#f19dc98f',
             fillOpacity: 1,
             strokeWeight: 2,
             clickable: true,
             editable: true,
             zIndex: 1
           }
         });
         drawingManager.setMap(this.map);
```

Sample Output



Ref:
https://developers.google.com/maps/documentation/javascript/examples/drawing-tools

7. **POI (Point of Interest)**

Contains information about a PointOfInterest that was clicked on.

```
public PointOfInterest(LatLng latLng, String placeId, String name)
//Create a POI
```

Parameters:

public final LatLng	latLng	The LatLng of the POI.
public final String	name	The name of the POI.
public final String	placeId	The placeId of the POI.

Reference:

https://developers.google.com/android/reference/com/google/android/gms/maps/model/Polyline

8. **To display information of a KML file**

```
var src = 'URL_for_KML_file';
var kmlLayer = new google.maps.KmlLayer(src, {
    suppressInfoWindows: true,
    preserveViewport: false,
    map: map
  });
kmlLayer.addListener('click', function(event) {
    var content = event.featureData.infoWindowHtml;
    var testimonial = document.getElementById('capture');
    testimonial.innerHTML = content;
  });
```

9. Agm Directions

`Agm-Direction` is the directive for `@agm/core`. Agm direction will give route from an origin location to destination location.

To use agm-direction install it using npm

```
npm i --save @agm/core agm-direction
```

Now import the modules in app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';

import { AgmCoreModule } from '@agm/core';            // @agm/core
import { AgmDirectionModule } from 'agm-direction';   //
agm-direction

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AgmCoreModule.forRoot({ // @agm/core
      apiKey: 'your key',
    }),
    AgmDirectionModule,     // agm-direction
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Usage: HTML:

```
<agm-map [latitude]="lat" [longitude]="lng">
  <agm-direction [origin]="origin" [destination]="destination">
  </agm-direction>
</agm-map>
```

TS:

```
public lat: Number = 24.799448;
public lng: Number = 120.979021;

public origin: any;
public destination: any;

ngOnInit() {
  this.getDirection();
}

getDirection() {
  this.origin = { lat: 24.799448, lng: 120.979021 };
  this.destination = { lat: 24.799524, lng: 120.975017 };

  // this.origin = 'Taipei Main Station';
  // this.destination = 'Taiwan Presidential Office';
}
```

MAPS STATIC API

The Maps Static API returns an image (either GIF, PNG or JPEG) in response to an HTTP request via a URL. For each request, you can specify the location of the map, the size of the image, the zoom level, the type of map, and the placement of optional markers at locations on the map. You can additionally label your markers using alphanumeric characters. A Maps Static API image is embedded within an `<img>` tag's `src` attribute, or its equivalent in other programming languages. If a

Maps Static API image is used outside of a web-based application (such as a browser) then a link must be included pointing to the displayed location in a web browser or native Google Maps application.

Ref: https://developers.google.com/maps/documentation/maps-static/dev-guide