1604C054 – Hybrid Mobile Programming

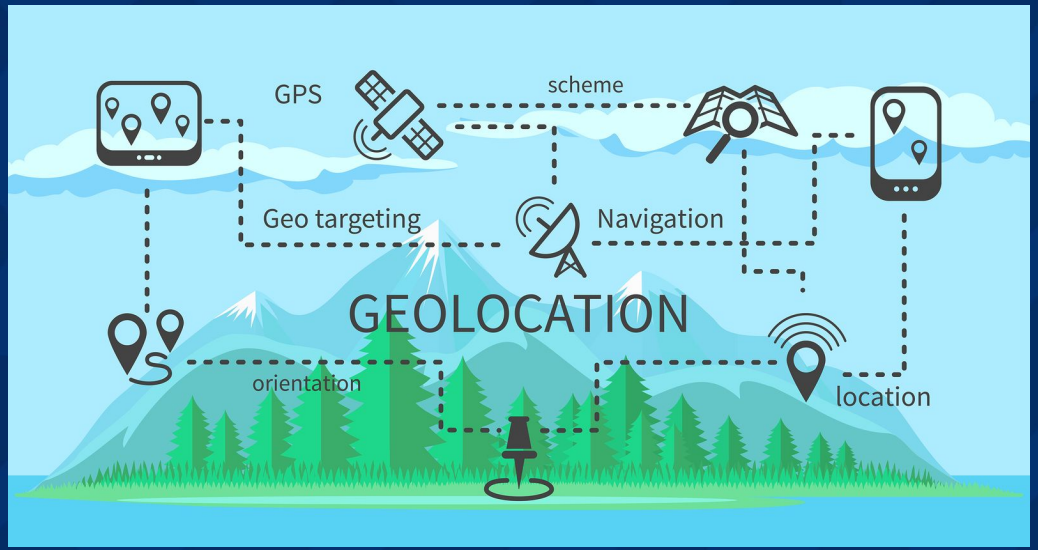
Location

WEEK 13

Informatics Engineering
Universitas Surabaya

Outline

1. Geolocation
2. Leaflet
3. Timer



Geolocation

What is Geolocation ?

- Geolocation refers to the process of determining the location or position of a person or device on Earth.
- This is typically done using a combination of satellite-based Global Positioning System (GPS), cellular network data, and Wi-Fi access points.
- Geolocation technology enables applications and services to provide specific information, services, or functionality based on the geographic coordinates (latitude and longitude) of a device.

Implementation

In PWA applications with javascript or typescript, geolocation capabilities can use browser capabilities. Example :

```
navigator.geolocation.getCurrentPosition((position) => {
  // Use position.coords.latitude and position.coords.longitude
     const latitude = position.coords.latitude;
     const longitude = position.coords.longitude; console.log('Latitude: ' + latitude);
     console.log('Longitude: ' + longitude);
 },
  (error) => {
    console.log('Error getting location', error);
  }
 );
```

Case Study #1

We will create a page to implement geolocation.

1. Create a new page named "location"

```
ionic generate page location
```

2. Prepare a link for this page in the drawer. On this page (app.component), we will display our location on the map.

```
<ion-menu-toggle>
        <ion-item routerLink="/location">
          <ion-icon name="location" slot="start"></ion-icon>
          <ion-label>Location</ion-label>
        </ion-item>
</ion-menu-toggle>
```

Get Location

In location.page.ts , prepare variables, functions, and function calls during ngOnInit.

```
lat:number=0
lon:number=0
```

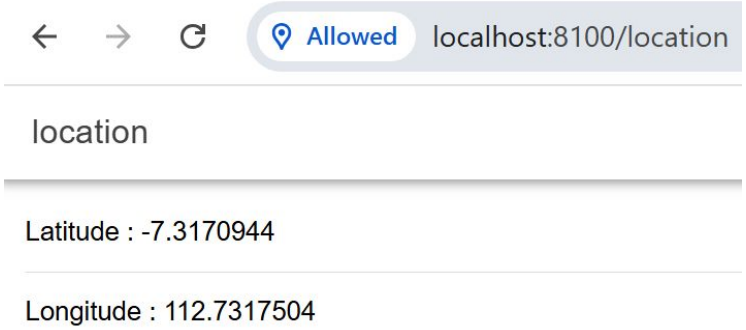
```
ngOnInit() {
    this.getCoordinates()
}
```

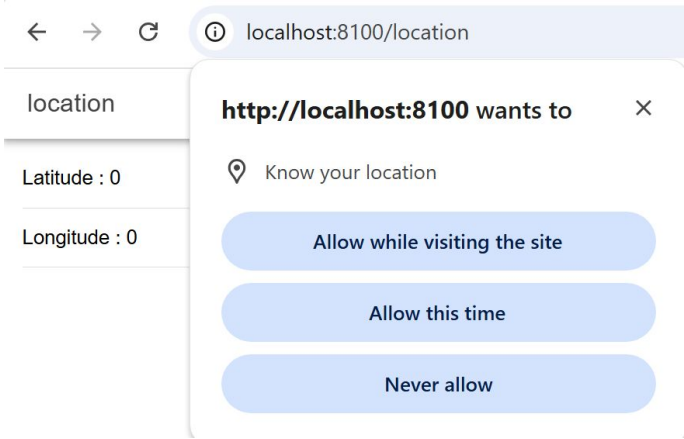
```
getCoordinates() {
    if (navigator.geolocation) {
      navigator.geolocation.getCurrentPosition((position) => {
        this.lat = position.coords.latitude;
        this.lon = position.coords.longitude;
      },
        (error) => {
          console.error('Error getting location', error);
        }
      );
    } else {
      console.error('Geolocation is not supported in this browser.');
    }
  }
```

Show - Lat and Long

In location.page.html

```
<ion-list>
   <ion-item> Latitude : {{lat}}
   </ion-item>
   <ion-item>
     Longitude : {{lon}}
   </ion-item>
 </ion-list>
```





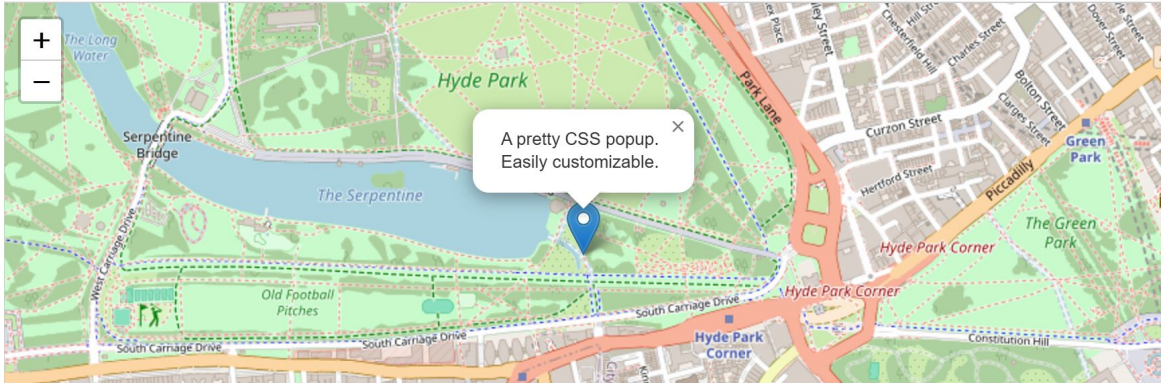
Do not choose ***never allow***

Leaflet

What is Leaflet?

- Leaflet is an open-source JavaScript library for interactive maps. It's designed to be lightweight, simple, and easy to use, making it a popular choice for developers who need to integrate maps into web applications.
- Leaflet provides a set of intuitive and customizable features for creating maps, handling user interactions, and displaying geographic information.

Website: leafletjs.com



Features of Leaflet

1. Lightweight and Modular: Leaflet is designed to be a lightweight library, making it easy to include in web projects. It consists of a core library and various plugins that can be added based on specific needs.
2. Map Display: Leaflet allows you to display interactive maps with support for different tile layers, including popular map providers like Googlemap, OpenStreetMap, Bing, Mapbox, and others. You can customize the map's appearance and behavior.
3. Markers and Popups: You can easily add markers to specific locations on the map, with the option to attach popups containing additional information. This is useful for highlighting points of interest or providing details about map features.

Features of Leaflet (2)

4. Layers and Overlays: Leaflet supports multiple layers, including base layers and overlays. This enables you to display different types of information on the map simultaneously, such as satellite imagery, terrain, or custom data overlays.
5. User Interaction: Leaflet provides support for various user interactions, including panning, zooming, and touch gestures. Users can explore the map seamlessly, and developers can customize the map's behavior based on user actions.
6. Extensibility: The Leaflet library is extensible, and developers can enhance its functionality by incorporating various plugins. These plugins cover a wide range of features, from heatmaps and clustering to drawing tools and geospatial analysis.
7. Cross-Browser Compatibility: Leaflet is designed to work across different web browsers, ensuring a consistent experience for users on various platforms
8. Open Source and Community Support: Leaflet is an open-source project with an active and supportive community. Developers can contribute to the project, report issues, and find resources, including documentation and tutorials.

Leaflet Installation

1. Install leaflet for your project

```
npm install leaflet
```

2. Install type of definition.

```
npm install --save-dev @types/leaflet
```

3. Import CSS in global.scss

```
@import "~leaflet/dist/leaflet.css";
```

4. Import Leaflet in specific page. ex : location.page

```
import * as L from 'leaflet';
```

if you get error add ***--force***

Add Map

The map can be placed on a Div with a specific id. This Div must be given a height and width size.

in location.page.html put it under lat lon

```
<ion-item>
        <div id="map"></div>
</ion-item>
```

In location.page.scss prepare the div size

```
#map{
        height: 400px;
        width: 100%;
}
```

Show Google Maps

In location.page.ts :

```
map: any;

initializeMap() {
   // Create a map centered at a specific location
   this.map = L.map('map').setView([this.lat, this.lon], 13);
   // Add a gmap street tile layer (you may use other providers, like bing OpenStreetMap, mapbox, etc.. )
   const googleStreets = L.tileLayer(
     'http://{s}.google.com/vt/lyrs=m&x={x}&y={y}&z={z}',
     { maxZoom: 20, subdomains: ['mt0', 'mt1', 'mt2', 'mt3'] }
   );
   googleStreets.addTo(this.map)
  }
```

Show Google Maps (2)

call ***initializeMap*** inside getCoordinates()

```
getCoordinates() {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition((position) => {
      this.lat = position.coords.latitude;
      this.lon = position.coords.longitude;
      this.initializeMap()
    },
      (error) => {
        console.error('Error getting location', error);
      }
    );
  } else {
    console.error('Geolocation is not supported in this browser.');
  }
}
```

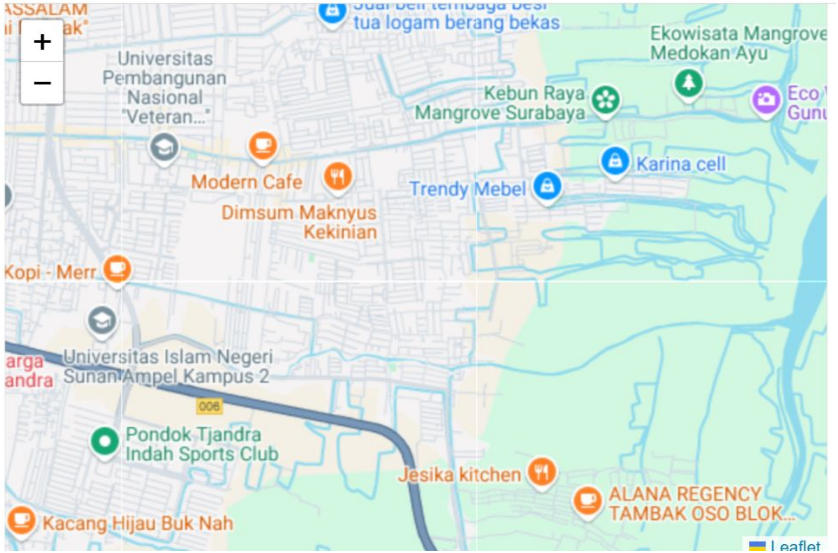
Show Google Maps (3)

See the result

location

Latitude : -7.3281053

Longitude : 112.8094962



Add Marker

Add variable in location.page :

```
markerLokasi:any;
```

Inside initializeMap function, after googleStreets consts, add:

```
var markerIcon = L.icon({
                                                                   iconUrl: 'https://toppng.com/uploads/preview/in-location-map-icon-navigation-symbol-ma-google-maps-marker-blue-11562916561qaf3tyejum.png', iconSize: [50, 50],
      iconAnchor: [25, 50],
    });
this.markerLokasi = L.marker([this.lat, this.lon], { icon: markerIcon })
this.markerLokasi.addTo(this.map);
```

nb: iconAnchor is the position of the image that shows the location of the point. the upper left part of the icon is 0.0 and the lower right is the width, height which in this icon the height and width are 25.25. The position is indicated by the sharp part of the icon in the bottom center, so the anchor value is 25.50

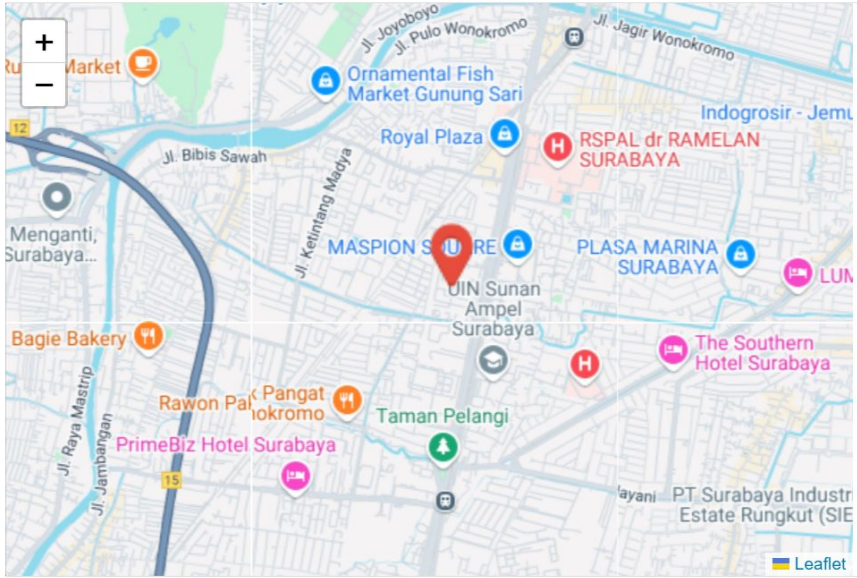
Add Marker (2)

See the result

location

Latitude : -7.3170944

Longitude : 112.7317504



Timer

Case Study #2

- We have been able to display our location on the map. Now, what if our position changes, for example while traveling and the application can display our latest position when we move locations.
- The simplest way is to update the location continuously at certain intervals. This can be done by using a timer that at each interval we will refresh the location.
- To save memory, we will not call initializeMap only once and in the next location retrieval only change the position of the marker and also the center of the map.

Interval Observable

Add in location.page :

```
import { interval, Subscription } from 'rxjs';
..

  timerSubscription: Subscription | undefined;
  isInit=false
  ...
  ...
  startTimer() {
     this.timerSubscription = interval(1000).subscribe(() => {
        this.getCoordinates()
    });
  }
```

Retrieving location every 1 second

Update getCoordinates()

initializeMap call on getCoordinates() is changed to:

```
if (!this.isInit) {
      this.initializeMap()
      this.isInit = true
      this.startTimer()
}
else {
     this.moving()
}
```

```
moving() {
  this.markerLokasi.setLatLng([this.lat,this.lon])
  this.map.flyTo([this.lat, this.lon], 13);
}
```

Try it on your device while walking or driving.

Case Study #3

- Next, we will try to display the location of friends/other people. Here, a movement simulation is carried out which can be taken from the web service: *https://ubaya.xyz/posisi_xy.php*
- We will continue to use the previous page, by adding another marker layer as a representation of the movement of friends.
- The movement simulation is carried out around Ubaya. so to check it we need to shift the map to Ubaya. For that, the change in the map center in the moving() function is temporarily deleted.

Prepare for Service

Although it is not very suitable, but to make it more concise, we put the API reading of position_xy in the foodservice.

```
position_xy(): Observable<any> {
    return this.http.get("https://ubaya.xyz/posisi_xy.php");
}
```

We will implement this function to location.page

Add layer marker

In location.page, add :

```
markerTeman:any;
lat2=0.0
lon2=0.0
```

inside initializeMap :

```
this.markerTeman=L.marker([this.lat2, this.lon2], {icon: markerIcon})
this.markerTeman.addTo(this.map);
```

Modify moving()

```
moving() {
    this.markerLokasi.setLatLng([this.lat, this.lon])
    // this.map.flyTo([this.lat, this.lon],13);
    this.foodservice.position_xy().subscribe((data) => {
      this.markerTeman.setLatLng([data.y, data.x])
    }
    );
  }
```

Do not forget to add foodservice declaration in constructor

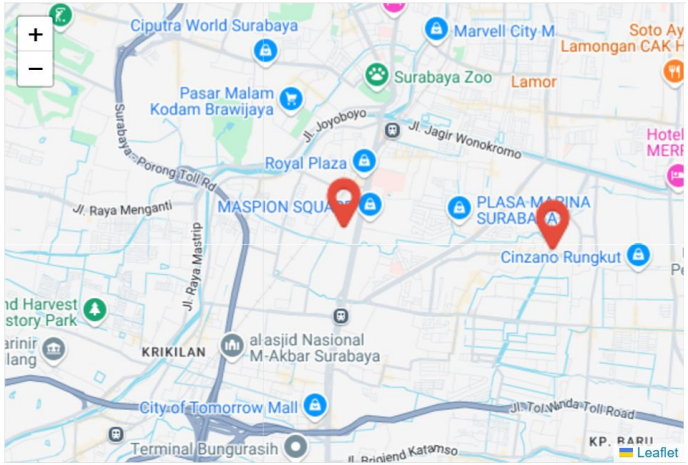
Result

Point the map at the road in front of Ubaya. You will see a marker that changes location every second.

location

Latitude : -7.3170944

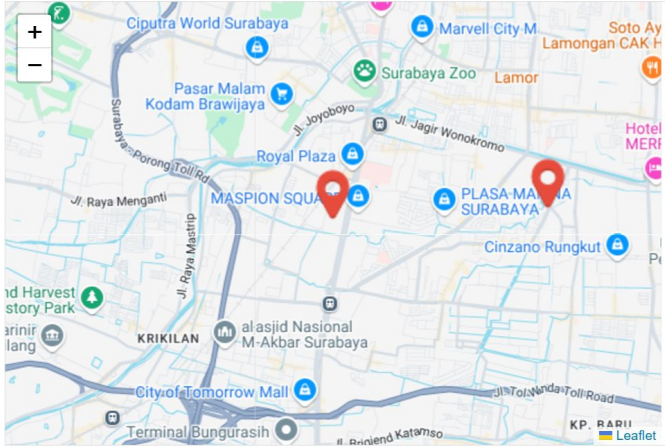
Longitude : 112.7317504



location

Latitude : -7.3170944

Longitude : 112.7317504



Exercise (One Week)

Can you replace the simulation API with real live location data from your moving friend?

Steps:

1. Create an API to update the location on the server. for example stored in a specific table.
2. Create an API to read this location
3. Add the use of the location update API at no. 1 every certain time range
4. Ask your friend to run the application on their device while walking/driving
5. Read the API at number 2 replacing the simulation API, run the application on your device. Then you will see the live location of your friend

Progress Project

Week 14 : Progress Project

- Progress Documentation (in .pdf)
- folder app
- package.json

Deadline : June 20, 2025

Thanks.

Any Question ?