Mobile Apps - Preparation - Leaflet Maps

Leaflet is an open-source JavaScript library for mobile-friendly interactive maps.

More info can be found at https://Leafletjs.com/.

OpenStreetMap is a collaborative project to create a free editable geographic database of the world. The geodata underlying the maps is considered the primary output of the project.

More info can be found at https://www.openstreetmap.org/

Basically while Leaflet is a framework for showing interactive maps, you still need a map provider. This is where openstreetmap comes in. Alternatives are for example GoogleMaps and Mapbox.

Leaflet dependency

Start by installing Leaflet as a dependency for your Vue3 project:

```
npm install Leaflet
```

At the moment of this writing, that would be Leaflet v1.9.3.

Setting up a Map

The first thing we need to do is setup a component with a container where the map can be mounted.

Let's start by creating a new component called `LeafletMap`.

```
<script setup lang="ts">
import { ref, onMounted } from 'vue';
import Leaflet from 'leaflet';

const center = ref([51.186917505979025, 3.2031807018500427] as
Leaflet.LatLngTuple)
const zoom = ref(13)    // Max = 19

const setup_leaflet = function() {
  const container = Leaflet.map("map_container").setView(center.value,
zoom.value);
  Leaflet.tileLayer(
    'https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png',
    {
      attribution: '&copy; <a
href="https://openstreetmap.org/copyright">OpenStreetMap contributors</a>',
      maxZoom: 19,
    }
  ).addTo(container);
}
```

```
onMounted(() => {
  console.log("Setting up the Leaflet map");
  setup_leaflet();
})
</script>

<template>
  <v-sheet class="text-center">
    <h1>Welcome to the wonderoes world of Leaflet maps</h1>
    <div id="map_container">
    </div>
  </v-sheet>
</template>

<style scoped>
#map_container {
  height: 50vh;
}
</style>
```

Important! Don't forget to add the import of the CSS file in the `main.ts` file. Otherwise the tiles will render all over the place and in the wrong order.

```
// ...
import "Leaflet/dist/Leaflet.css";
// ...
```

Next make sure to display the component somewhere and allow access to it via a route.

Passing Location for Markers

If we want to indicated something on a map, we need to place some kind of markers. Let's pass some locations to the map component.

```
<script setup lang="ts">
import LeafletMap from '@/components/LeafletMap.vue'
import { ref } from 'vue';
import type { Location } from '@/types/location';

const locations = ref([
  { id: '1', name: 'VIVES Xaverianen', lat: 51.18721897883223, lng:
3.2029320966172814 },
  { id: '2', name: 'Frituur Bosrand', lat: 51.187986416742454, lng:
3.209243333778497 },
  { id: '3', name: 'VIVES Station', lat: 51.1940399495556, lng:
3.2180740271685564 },
] as Location[]);
</script>
```

```
<template>
  <v-app>
    <v-main>
      <Leaflet-map :locations="locations" />
    </v-main>
  </v-app>
</template>
```

Where `location.ts` contains

```
export interface Location {
  id: string;
  name: string;
  lat: number;
  lng: number;
}
```

Now we can take in the `locations` as a prop:

```
// ...

import type { PropType } from 'vue';
import type { Location } from '@/types/location';

const props = defineProps({
  locations: { type: Array as PropType<Location[]>, default: () => [] }
})
```

Now we can create markers and show them on the map. We can even add an event handler in case someone clicks them:

```
const setup_leaflet = function() {
  // ...

  // Add markers
  props.locations.forEach((loc) => {
    Leaflet.marker([loc.lat,
loc.lng]).bindPopup(loc.name).addTo(container).on('click', (e) => {
      console.log("Someone clicked the marker at " + e.latlng);
    });
  })
}
```

However we have a problem here. There is no efficient way to determine what actual `location` was clicked as we do not have a way to bind our location object to the marker.

Extending the Marker class

As encouraged by any OOP or software enthousiast, inheritance is our saviour here. We can extend the actual `Leaflet.Marker` class and implement our own data attribute to store more info.

Let's create a `DataMarker.ts`:

```
import Leaflet from 'leaflet';

export class DataMarker extends Leaflet.Marker {
  data: any;

  constructor(latLng: Leaflet.LatLngExpression, data: any, options?:
Leaflet.MarkerOptions) {
    super(latLng, options);
    this.data = data;
  }
}
```

Now we need to add our own objects to the map instead of the default markers:

```
//...
import Leaflet from 'leaflet';
import { DataMarker } from '@/lib/DataMarker';

const setup_leaflet = function() {
  // ...

  // Add markers
  props.locations.forEach((loc) => {
    (new DataMarker([loc.lat, loc.lng],
loc)).bindPopup(loc.name).addTo(container).on('click', (e) => {
      console.log("Someone clicked the marker at " + e.latlng);
      console.log(e.target.data)
    });
  })
}
```

Not Reactive

This solution is not reactive as the markers are populated when the map is mounted. You can test this by adding a button below the map that adds another location to the list.

```
<script setup lang="ts">
import LeafletMap from '@/components/LeafletMap.vue'
import { ref } from 'vue';
import type { Location } from '@/types/location';
```

```
const locations = ref([
  { id: '1', name: 'VIVES Xaverianen', lat: 51.18721897883223, lng:
3.2029320966172814 },
  { id: '2', name: 'Frituur Bosrand', lat: 51.187986416742454, lng:
3.209243333778497 },
  { id: '3', name: 'VIVES Station', lat: 51.1940399495556, lng:
3.2180740271685564 },
] as Location[]);

const add_location = () => {
  locations.value.push({
    id: `${locations.value.length+1}`,
    name: 'Who Knows',
    lat: locations.value[locations.value.length-1].lat + 0.005,
    lng: locations.value[locations.value.length-1].lng + 0.005
  });

  console.log(locations.value)
}
</script>

<template>
  <v-app>
    <v-main>
      <leaflet-map :locations="locations" />

      <v-card>
        <v-card-title>Add a location</v-card-title>
        <v-card-actions>
          <v-btn color="primary" @click="add_location">Add Location</v-btn>
        </v-card-actions>
      </v-card>
    </v-main>
  </v-app>
</template>
```

Using a Watcher

We need to watch the locations in the `LeafletMap.vue` component. When it changes we have to remove the markers and create the new ones again. This will require some refactoring.

1. First we need to save the map instance to an object so we can change it later on

```
// ...

const map = {
  container: {} as Leaflet.Map,
}

const setup_leaflet = function() {
  map.container = Leaflet.map("map_container").setView(center.value,
zoom.value);
  Leaflet.tileLayer(
    'https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png',
    {
      attribution: '&copy; <a
href="https://openstreetmap.org/copyright">OpenStreetMap contributors</a>',
      maxZoom: 19,
    }
  ).addTo(map.container);

  // ...
}
```

2. Next we need to combine the markers into a Leaflet `layerGroup`. This can then we added to the map and also removed.

```
// ...

const map = {
  container: {} as Leaflet.Map,
  markers: {} as Leaflet.LayerGroup,
}

const setup_leaflet = function() {
  // ...

  // Create layer with markers
  map.markers = Leaflet.layerGroup();

  // Add markers to the layer group
  props.locations.forEach((loc) => {
    (new DataMarker([loc.lat, loc.lng],
loc)).bindPopup(loc.name).addTo(map.markers).on('click', (e) => {
      console.log("Someone clicked the marker at " + e.latlng);
      console.log(e.target.data)
    });
  });

  // Add the layer to the map
  map.markers.addTo(map.container);
}
```

3. Now if we wish to clear the markers and add them anew using a watcher, we should refactor the `setup_leaflet()` function and extract the marker creation into its own method.

```
const setup_leaflet = function() {
  map.container = Leaflet.map("map_container").setView(center.value,
zoom.value);
  Leaflet.tileLayer(
    'https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png',
    {
      attribution: '&copy; <a
href="https://openstreetmap.org/copyright">OpenStreetMap contributors</a>',
      maxZoom: 19,
    }
  ).addTo(map.container);

  add_markers_to_map();
}

const add_markers_to_map = function() {
  // Create layer with markers
  map.markers = Leaflet.layerGroup();

  // Add markers to the layer group
  props.locations.forEach((loc) => {
    (new DataMarker([loc.lat, loc.lng],
loc)).bindPopup(loc.name).addTo(map.markers).on('click', (e) => {
      console.log("Someone clicked the marker at " + e.latlng);
      console.log(e.target.data)
    });
  });

  // Add the layer to the map
  map.markers.addTo(map.container);
}
```

4. We should also remove the marker layer from the map before creating a new one. Otherwise we will get layer on top of layer

```
const add_markers_to_map = function() {
  // Remove old layer of markers
  if (map.markers) map.container.removeLayer(map.markers)

  // Create layer with markers
  map.markers = Leaflet.layerGroup();

  // ...
}
```

And now we can setup a simple watcher for the locations that calls the `add_markers_to_map()` when locations is changed. It's best to setup the watcher in the `setup_leaflet` method.

```
const setup_leaflet = function() {
  // ...

  // Renew the markers when locations change
  watch(
    () => props.locations,     // Can't watch property of reactive object
    (locations) => {
      console.log('Locations changed')
      add_markers_to_map();
    },
    { deep: true }      // Force deep watcher
  )
}
```

Not sure if this is the best option here. Deep watchers are expensive, especially on larger collections.

Another Option

Another option may be found in some sort of *dirty* variable that indicates that the map needs to refresh.

```
//...

const props = defineProps({
  locations: { type: Array as PropType<Location[]>, default: () => [] },
  refreshTicker: { type: Number, default: 0 }
})

//...

const setup_leaflet = function() {
  // ....

  watch(
    () => props.refreshTicker,     // Can't watch property of reactive
object
    (refreshTicker) => {
      console.log('Refresh of map markers required')
      add_markers_to_map();
    },
  )
}
```

Now we can just use the `refreshTicker` as some sort of trigger.

```
<script setup lang="ts">
// ...

const refreshTicker = ref(0);
const add_location = () => {
  // ...

  console.log(locations.value)
  refreshTicker.value++;
}
```

```
</script>

<template>
  <v-app>
    <v-main>
      <leaflet-map :locations="locations" :refreshTicker="refreshTicker" />

      <v-card>
        <v-card-title>Add a location</v-card-title>
        <v-card-actions>
          <v-btn color="primary" @click="add_location">Add Location</v-btn>
        </v-card-actions>
      </v-card>
    </v-main>
  </v-app>
</template>
```