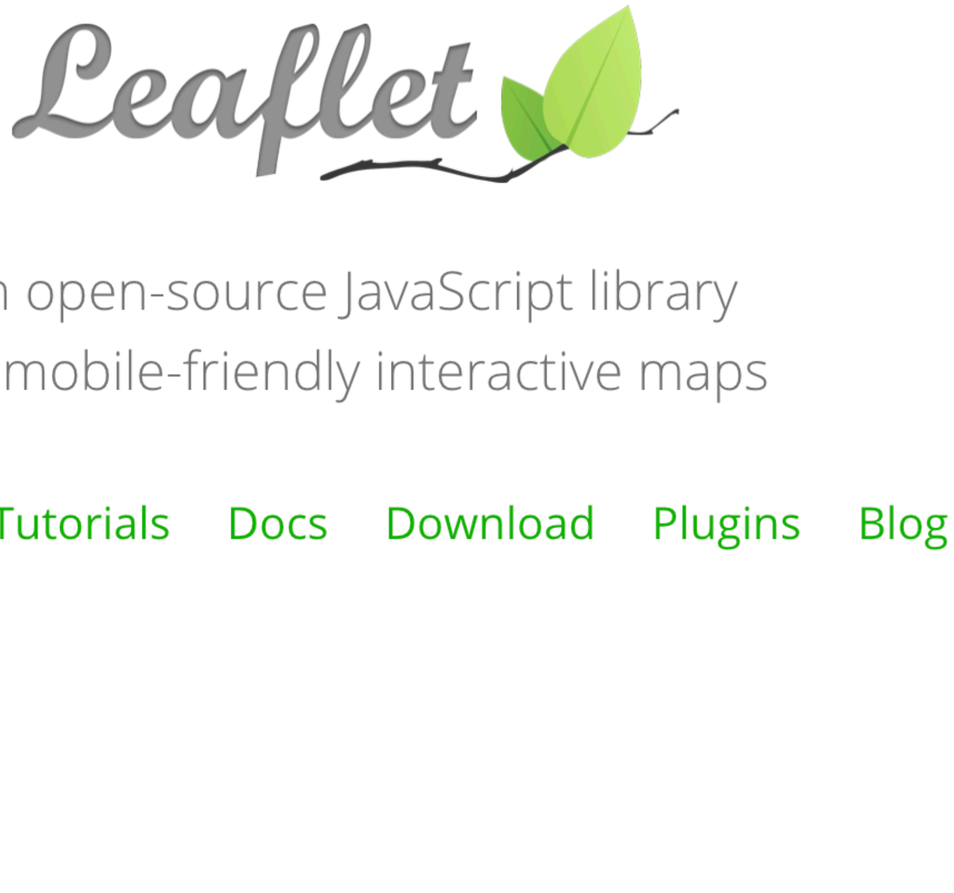
Leaflet Maps

Introducing to using Leaflet maps in Svelte applications

https://leafletjs.com/



- Leaflet is a widely used open source JavaScript library used to build web mapping applications. First released in 2011 it supports most mobile and desktop platforms, supporting HTML5 and CSS3. Along with OpenLayers, and the Google Maps API, it is one of the most popular JavaScript mapping libraries and is used by major web sites such as FourSquare, Pinterest and Flickr.

- Leaflet allows developers without a GIS background to very easily display tiled web maps hosted on a public server, with optional tiled overlays. It can load feature data from GeoJSON files, style it and create interactive layers, such as markers with popups when clicked.

Map

Usage example
Creation
Options
Events

Map Methods

Modifying map state
Getting map state
Layers and controls
Conversion methods
Other methods

Map Misc

Properties
Panes

UI Layers

Marker
Popup
Tooltip

Raster Layers

TileLayer
TileLayer.WMS
ImageOverlay
VideoOverlay

Vector Layers

Path
Polyline
Polygon
Rectangle
Circle
CircleMarker
SVG
Canvas

Other Layers

LayerGroup
FeatureGroup
GeoJSON
GridLayer

Basic Types

LatLng
LatLngBounds
Point
Bounds
Icon
DivIcon

Controls

Zoom
Attribution
Layers
Scale

Utility

Browser
Util
Transformation
LineUtil
PolyUtil

DOM Utility

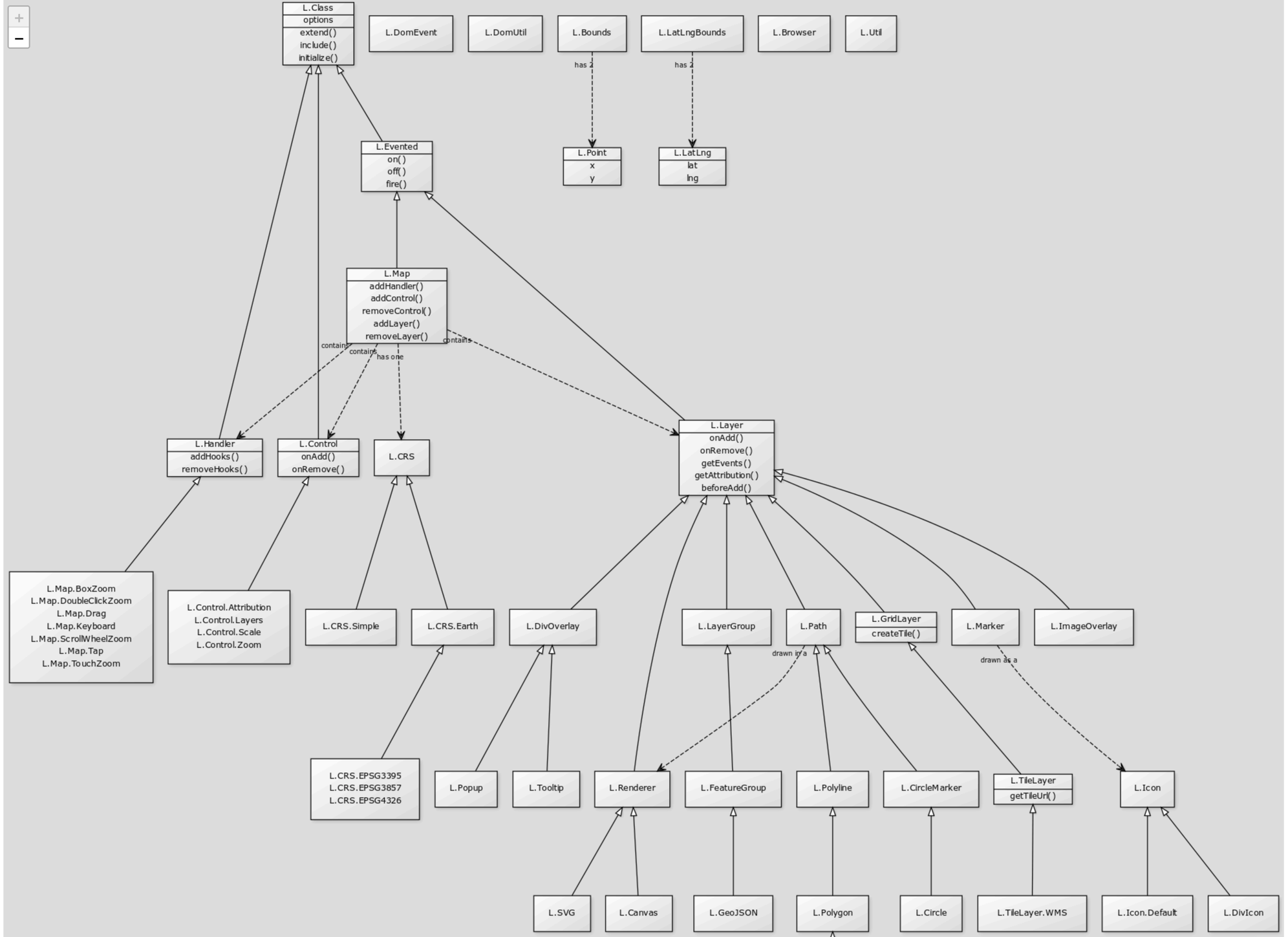
DomEvent
DomUtil
PosAnimation
Draggable

Base Classes

Class
Evented
Layer
Interactive layer
Control
Handler
Projection
CRS
Renderer

Misc

Event objects
global switches
noConflict
version



Leaflet Javascript Object Model

Leaflet Plugins

While Leaflet is meant to be as lightweight as possible, and focuses on a core set of features, an easy way to extend its functionality is to use third-party plugins. Thanks to the awesome community behind Leaflet, there are literally hundreds of nice plugins to choose from.

Tile & image layers

Basemap providers
Basemap formats
Non-map base layers
Tile/image display
Tile load
Vector tiles

Overlay data

Overlay data formats
Dynamic data loading
Synthetic overlays
Data providers

Overlay Display

Markers & renderers
Overlay animations
Clustering/decluttering
Heatmaps
DataViz

Overlay interaction

Edit geometries
Time & elevation
Search & popups
Area/overlay selection

Map interaction

Layer switching controls
Interactive pan/zoom
Bookmarked pan/zoom
Fullscreen
Minimaps & synced maps
Measurement
Mouse coordinates
Events
User interface
Print/export
Geolocation

Miscellaneous

Geoprocessing
Routing
Geocoding
Plugin collections

Integration

Frameworks & build systems
3rd party

Develop your own

Leaflet in Svelte

- Download and make available the leaflet javascript library to the Svelte application components


```
npm install -D leaflet
npm install -D @types/leaflet
```

```
"devDependencies": {
  "@sveltejs/adapter-auto": "^3.2.0",
  "@sveltejs/kit": "^2.5.5",
  "@sveltejs/vite-plugin-svelte": "^3.0.2",
  "@types/eslint": "^8.56.7",
  "@types/leaflet": "^1.9.9",
  "@typescript-eslint/eslint-plugin": "^7.6.0",
  "@typescript-eslint/parser": "^7.6.0",
  "axios": "^1.6.8",
  "eslint": "^8.56.0",
  "eslint-config-prettier": "^9.1.0",
  "eslint-plugin-svelte": "^2.36.0",
  "leaflet": "^1.9.4",
  "prettier": "^3.2.5",
  "prettier-plugin-svelte": "^3.2.2",
  "svelte": "^4.2.12",
  "svelte-check": "^3.6.9",
  "svelte-frappe-charts": "^1.10.0",
  "tslib": "^2.6.2",
  "typescript": "^5.4.4",
  "vite": "^5.2.8"
},
```

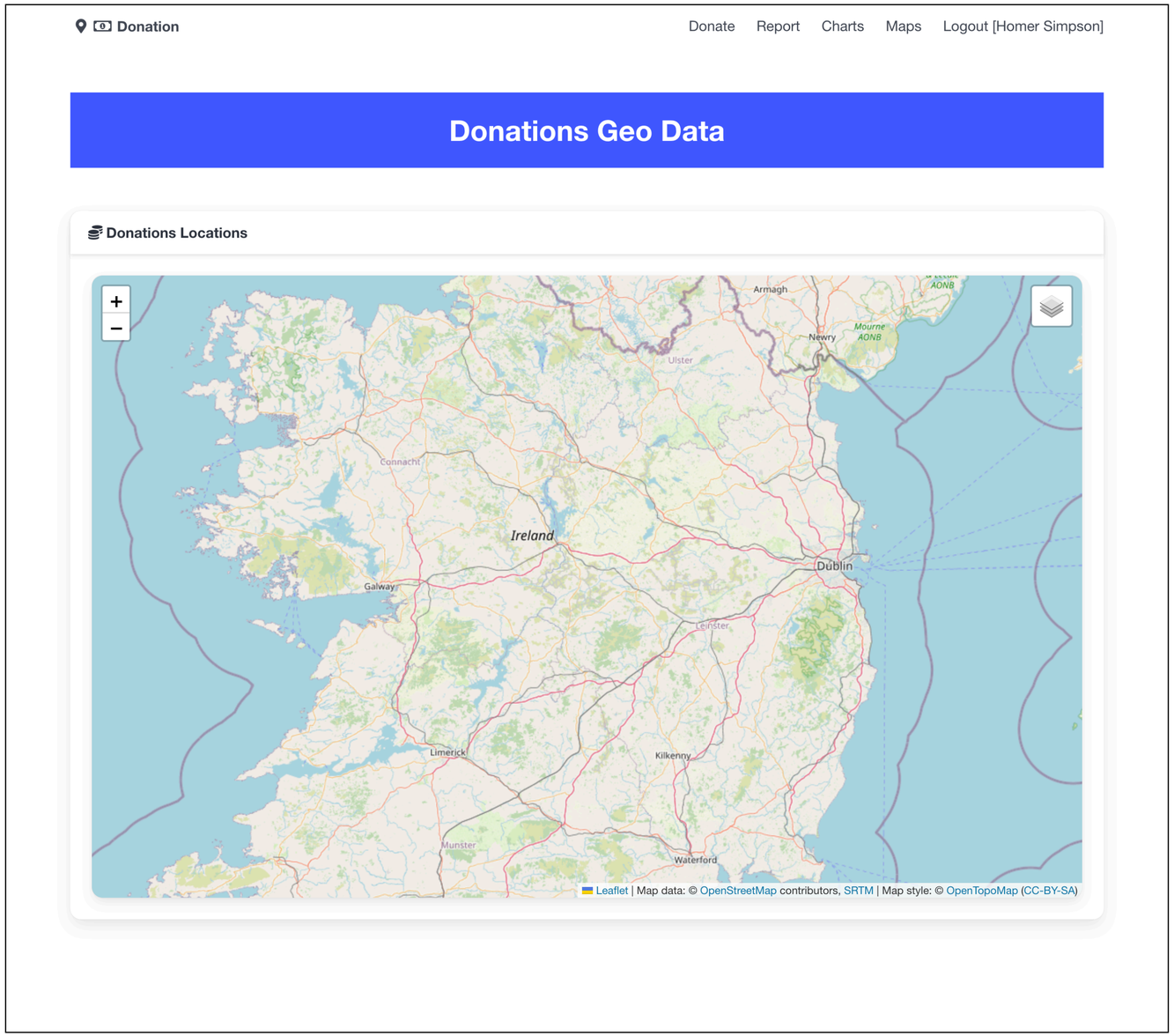
Wrapper Class - LeafletMap

- Developed to simplify Leaflet use.

```svelte
<script lang="ts">
  import "leaflet/dist/leaflet.css";
  import { onMount } from "svelte";
  import type { Control, Map as LeafletMap } from "leaflet";

  export let id = "home-map-id";
  export let height = 80;
  export let location = { lat: 53.2734, lng: -7.7783203 };
  export let zoom = 8;
  export let minZoom = 7;
  export let activeLayer = "Terrain";

  let imap: LeafletMap;
  let control: Control.Layers;
  let overlays: Control.LayersObject = {};
  let baseLayers: any;

  onMount(async () => {
    const leaflet = await import("leaflet");
    baseLayers = {
      Terrain: leaflet.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
        maxZoom: 17,
        attribution:
          'Map data: &copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
      }),
      Satellite: leaflet.tileLayer("https://server.arcgisonline.com/ArcGIS/rest/services/
        attribution: "Tiles &copy; Esri &mdash; Source: Esri, i-cubed, USDA, USGS, AEX, G
      })
    };
    let defaultLayer = baseLayers[activeLayer];
    imap = leaflet.map(id, {
      center: [location.lat, location.lng],
      zoom: zoom,
      minZoom: minZoom,
      layers: [defaultLayer]
    });
    control = leaflet.control.layers(baseLayers, overlays).addTo(imap);
  });
</script>

<div {id} class="box" style="height: {height}vh" />
```

LeafletMap.svelte

Map Route



routes/maps/+page.svelte

```svelte
<script lang="ts">
  import { subTitle } from "$lib/stores";
  import Card from "$lib/ui/Card.svelte";
  import LeafletMap from "$lib/ui/LeafletMap.svelte";

  subTitle.set("Donations Geo Data");
</script>

<Card title="Donations Locations">
  <LeafletMap height={60} />
</Card>
```

Wrapper Class - LeafletMap

- When the component is mounted onto the DOM
  - Import the leaflet component
  - Create the ‘base layers’
    - Terrain
    - Satellite
  - Create the layers
  - Create the leaflet map objects with a set of default values
  - Add the layer control

```
onMount(async () => {
  const leaflet = await import("leaflet");
  baseLayers = {
    Terrain: leaflet.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
      maxZoom: 17,
      attribution:
        'Map data: &copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
    }),
    Satellite: leaflet.tileLayer("https://server.arcgisonline.com/ArcGIS/rest/services/
      attribution: "Tiles &copy; Esri &mdash; Source: Esri, i-cubed, USDA, USGS, AEX, Ge
    })
  };
  let defaultLayer = baseLayers[activeLayer];
  imap = leaflet.map(id, {
    center: [location.lat, location.lng],
    zoom: zoom,
    minZoom: minZoom,
    layers: [defaultLayer]
  });
  control = leaflet.control.layers(baseLayers, overlays).addTo(imap);
});
```

LeafletMap.svelte

Wrapper Class - LeafletMap

- When the component is mounted onto the DOM
  - Import the leaflet component
  - Create the ‘base layers’
    - Terrain
    - Satellite
  - Create the layers
  - Create the leaflet map objects with a set of default values
  - Add the layer control

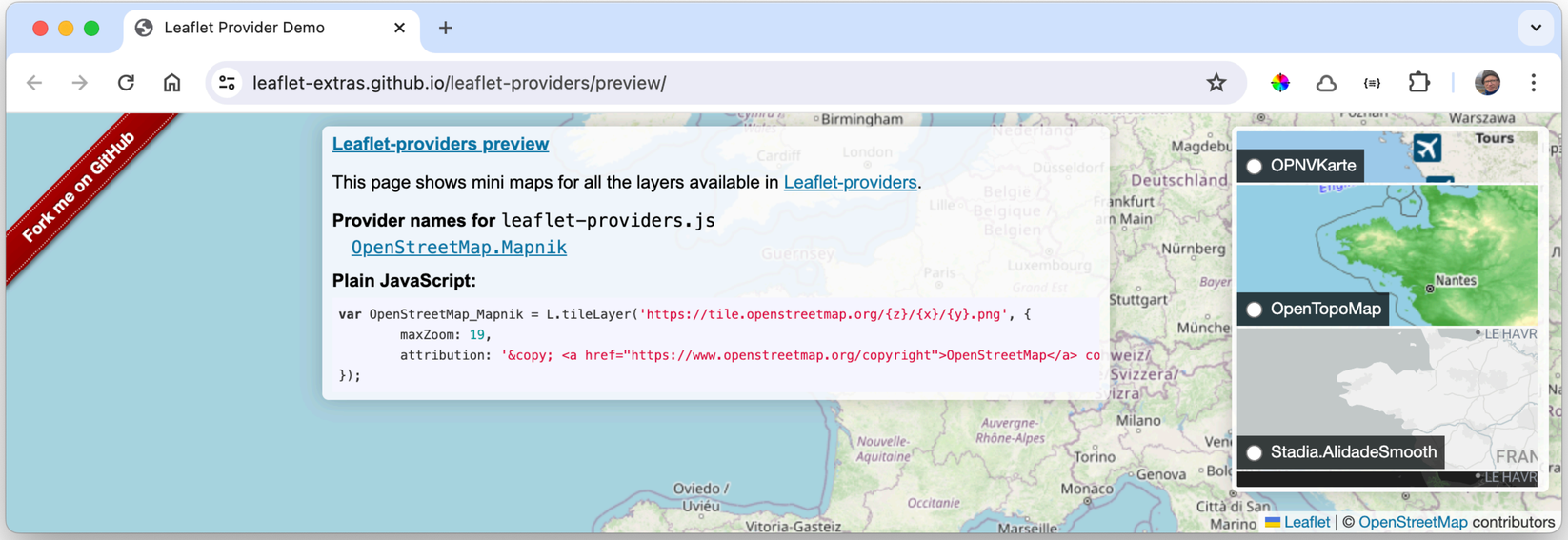
```
onMount(async () => {
  const leaflet = await import("leaflet");
  baseLayers = {
    Terrain: leaflet.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
      maxZoom: 17,
      attribution:
        'Map data: &copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
    }),
    Satellite: leaflet.tileLayer("https://server.arcgisonline.com/ArcGIS/rest/services/
      attribution: "Tiles &copy; Esri &mdash; Source: Esri, i-cubed, USDA, USGS, AEX, Ge
    })
  };
  let defaultLayer = baseLayers[activeLayer];
  imap = leaflet.map(id, {
    center: [location.lat, location.lng],
    zoom: zoom,
    minZoom: minZoom,
    layers: [defaultLayer]
  });
  control = leaflet.control.layers(baseLayers, overlays).addTo(imap);
});
```

LeafletMap.svelte

Layers

```
baseLayers = {
  Terrain: leaflet.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
    maxZoom: 17,
    attribution:
      'Map data: &copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
  }),
  Satellite: leaflet.tileLayer("https://server.arcgisonline.com/ArcGIS/rest/services/
    attribution: "Tiles &copy; Esri &mdash; Source: Esri, i-cubed, USDA, USGS, AEX, G
  })
```

- Terrain
- Satellite



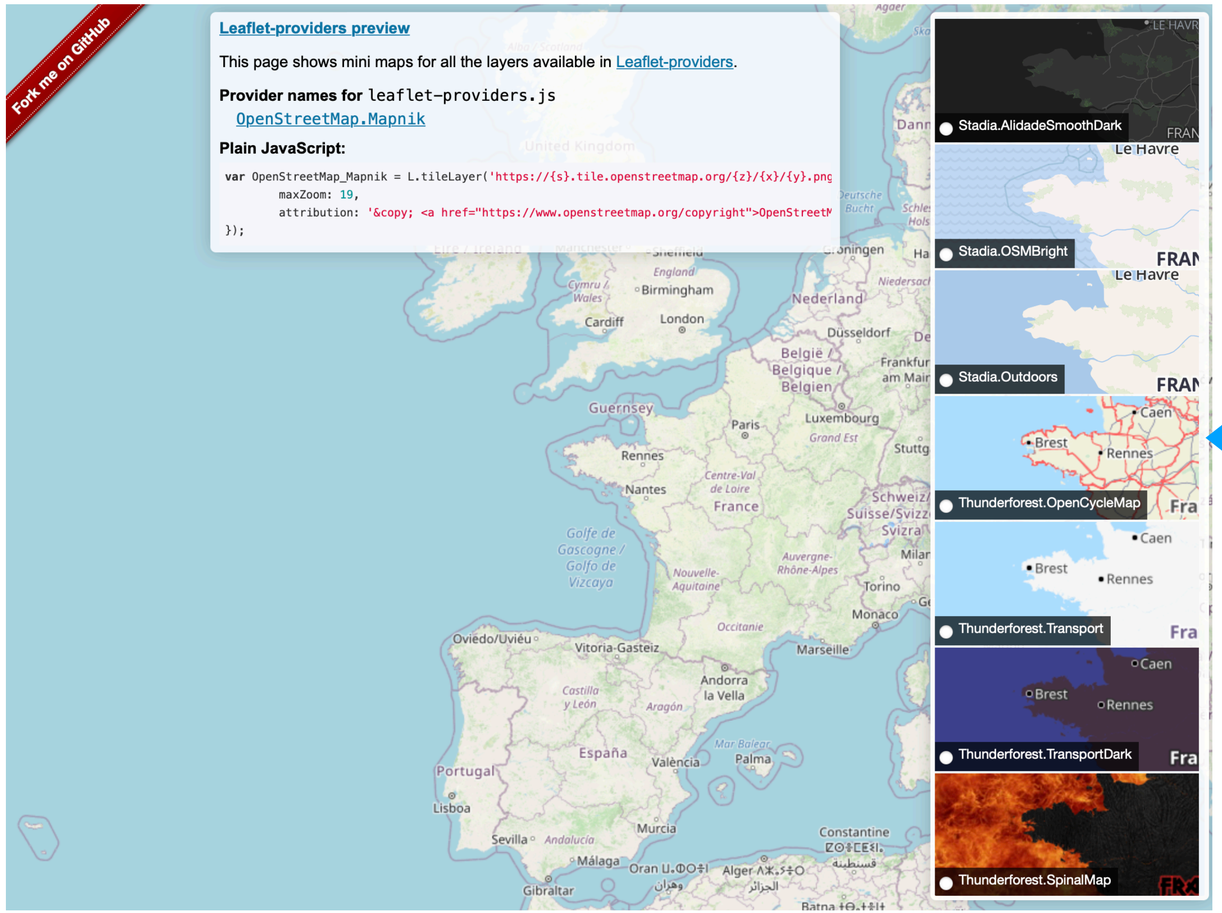
- Open Repository of layers

https://leaflet-extras.github.io/leaflet-providers/preview/

Leaflet Layers

- Defined in leaflet-map.js wrapper

```
baseLayers = {
  Terrain: L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
    maxZoom: 17,
    attribution:
      'Map data: &copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors, <a hr
  }),
  Satellite: L.tileLayer("https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}
    attribution:
      "Tiles &copy; Esri &mdash; Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGF
  }),
};
```

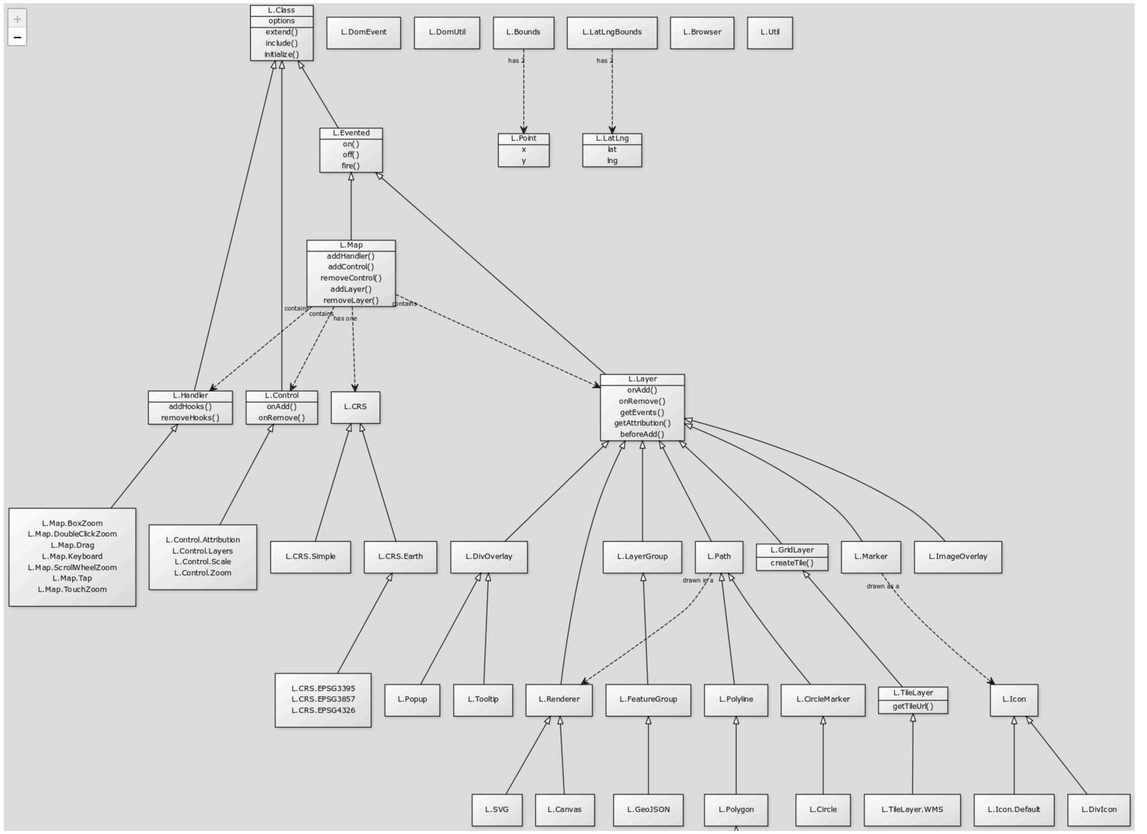


https://github.com/leaflet-extras/leaflet-providers

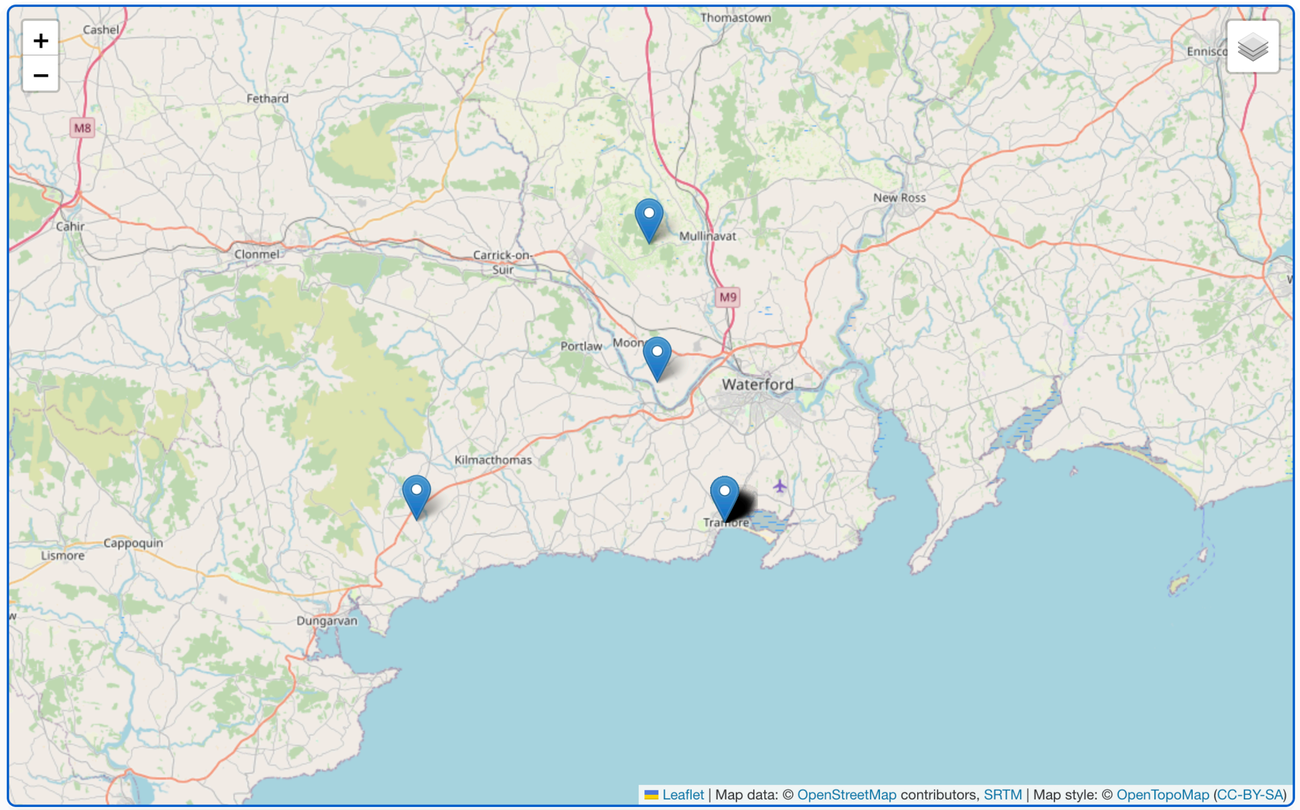
- Preview other layers
- Some require registration/API keys

LeafletMap Methods

- addLayer(title, layer)
- addLayerGroup(title)
- showLayerControl()
- showZoomControl()
- moveTo(zoom, location)
- addMarker(location, popupText, layerTitle)



Donations Locations



- For each donation, place a pin on the map and the donation's location.

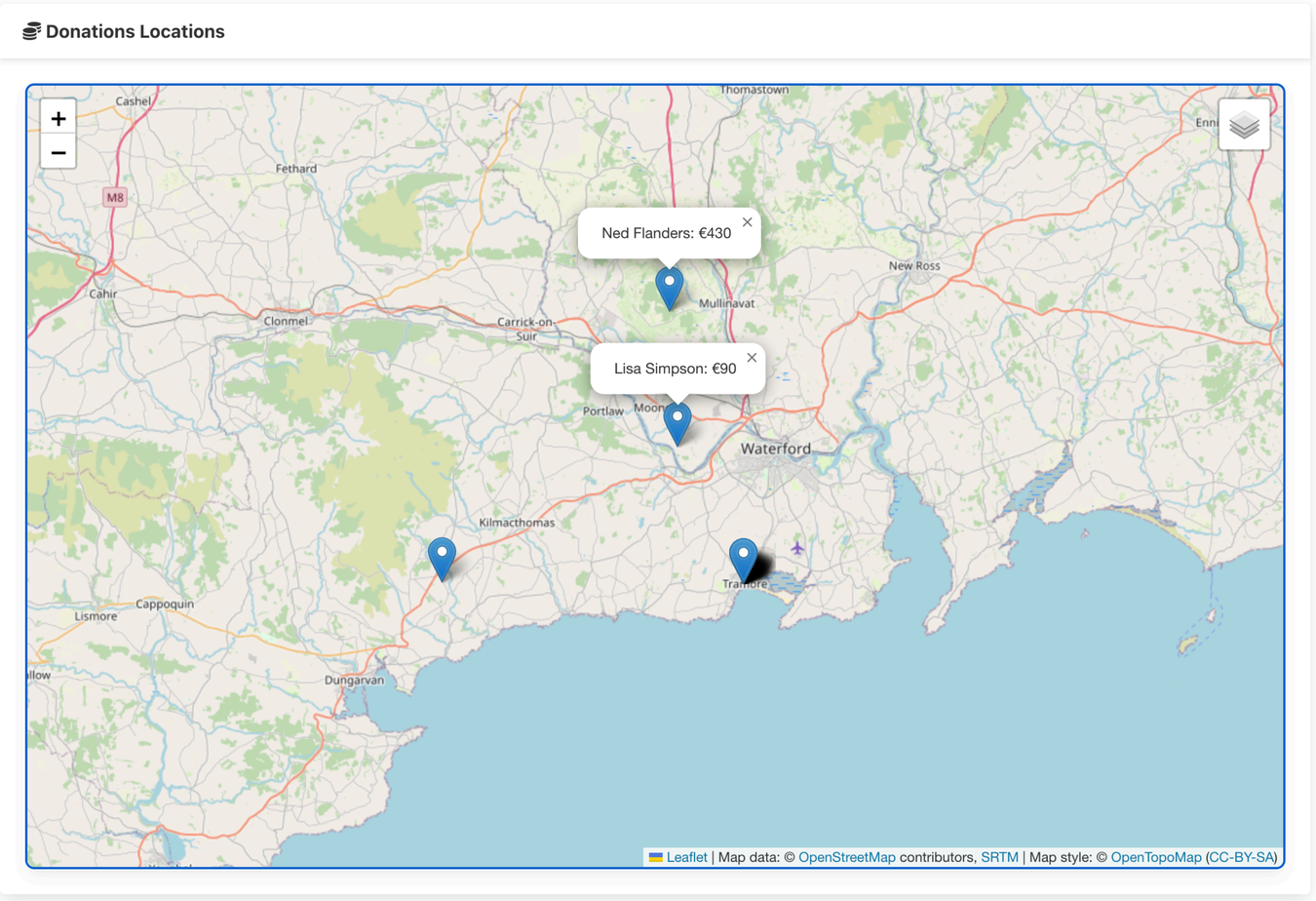
- Extend LeafletMap to include a new function:

```
export function addMarker(lat: number, lng: number)
  const marker = L.marker([lat, lng]).addTo(imap);
}
```

- In the Maps route, retrieve the donations and add the markers :

Markers

```
onMount(async () => {
  const donations = await donationService.getDonations(get(currentSession));
  donations.forEach((donation: Donation) => {
    map.addMarker(donation.lat, donation.lng);
  });
});
```



- If a pin is selected, display a popup with custom text

- Extend addMarker to use the appropriate Leafler API:

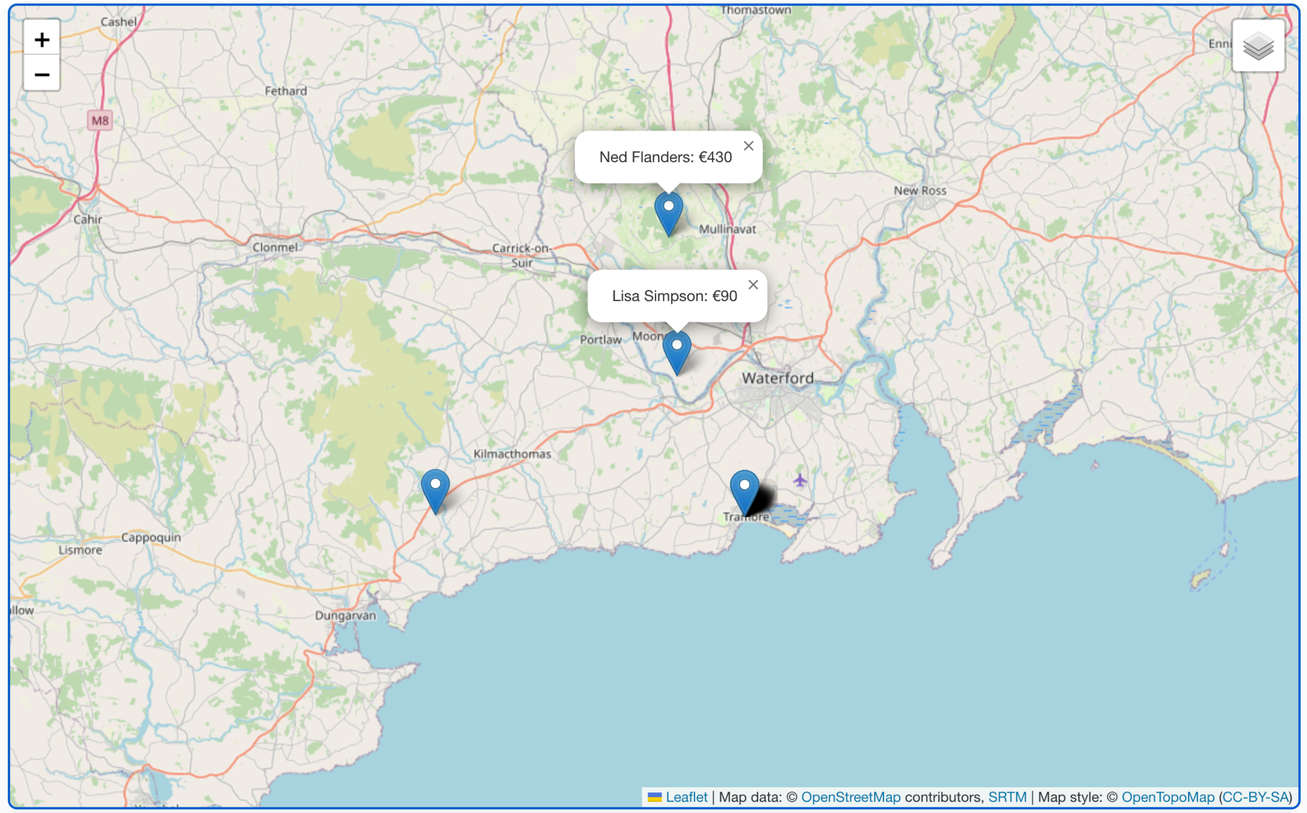
```
export function addMarker(lat: number, lng: number, popupText: string) {
  const marker = L.marker([lat, lng]).addTo(imap);
  const popup = L.popup({ autoClose: false, closeOnClick: false });
  popup.setContent(popupText);
  marker.bindPopup(popup);
}
```

- In the Maps route, prepare the popup text & pass the addMarker:

Popups

```
onMount(async () => {
  const donations = await donationService.getDonations(get(currentSession));
  donations.forEach((donation: Donation) => {
    if (typeof donation.candidate !== "string") {
      const popup = `${donation.candidate.firstName} ${donation.candidate.lastName}: €${donation.amount}`;
      map.addMarker(donation.lat, donation.lng, popup);
    }
  });
});
```

Donations Locations



- Pan and Zoom to a specific location on a map.

- In LeafletMap, introduce new function to support this:

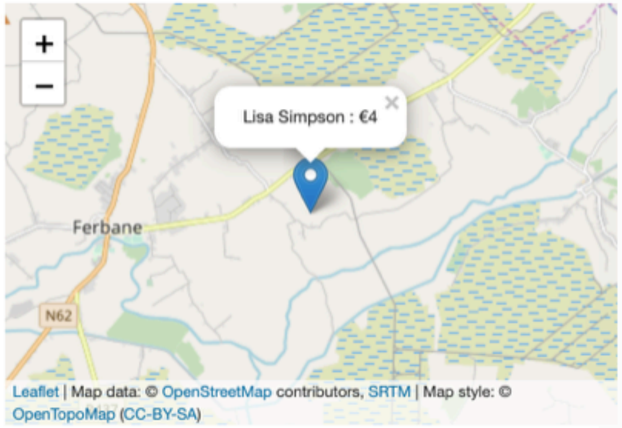
```
export function moveTo(lat: number, lng: number) {
  imap.flyTo({ lat: lat, lng: lng });
}
```

- In the Maps route onMount, move to the last donation added:

MoveTo

```
onMount(async () => {
  const donations = await donationService.getDonations(get(currentSession));
  donations.forEach((donation: Donation) => {
    if (typeof donation.candidate !== "string") {
      const popup = `${donation.candidate.firstName} ${donation.candidate.lastName}: €${donation.amount}`;
      map.addMarker(donation.lat, donation.lng, popup);
    }
  });
  const lastDonation = donations[donations.length - 1];
  if (lastDonation) map.moveTo(lastDonation.lat, lastDonation.lng);
});
```

Leaflet Maps



Introducing to using Leaflet maps in Svelte applications