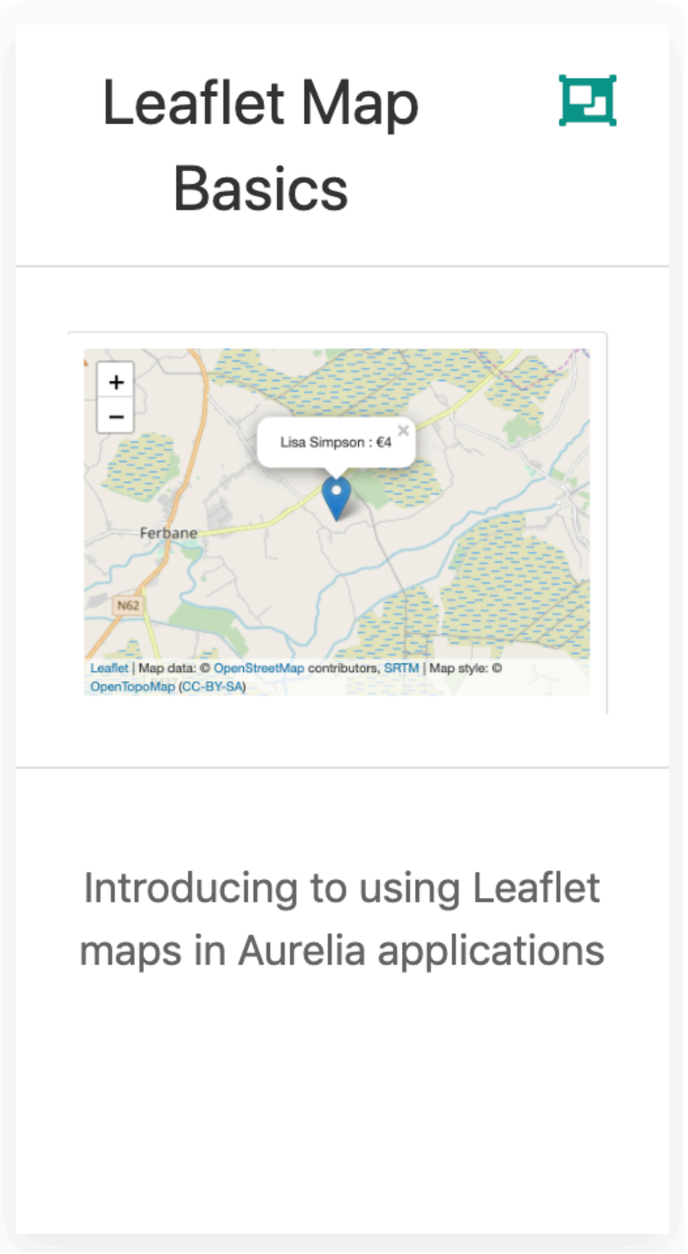
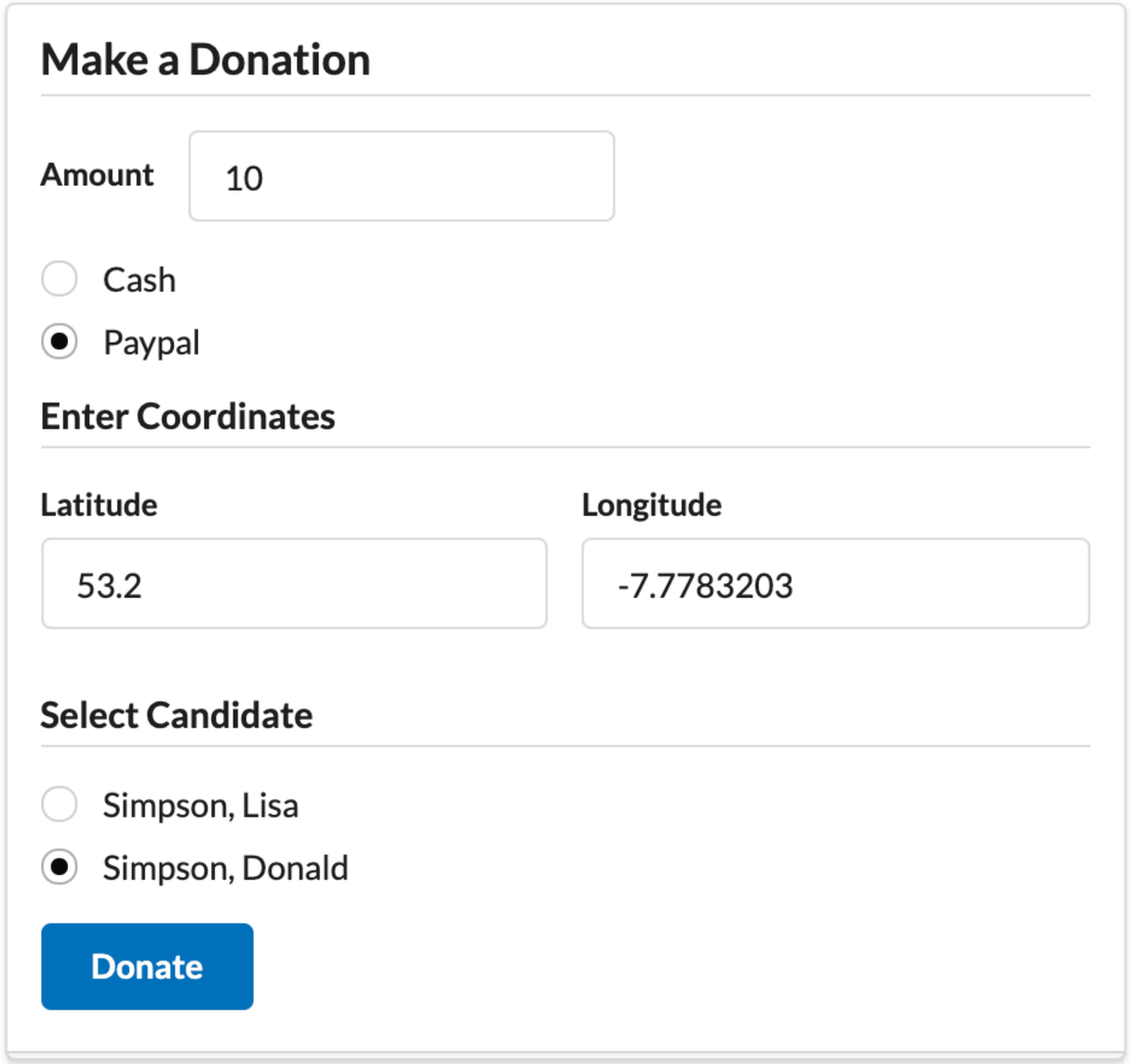
Leaflet Basics



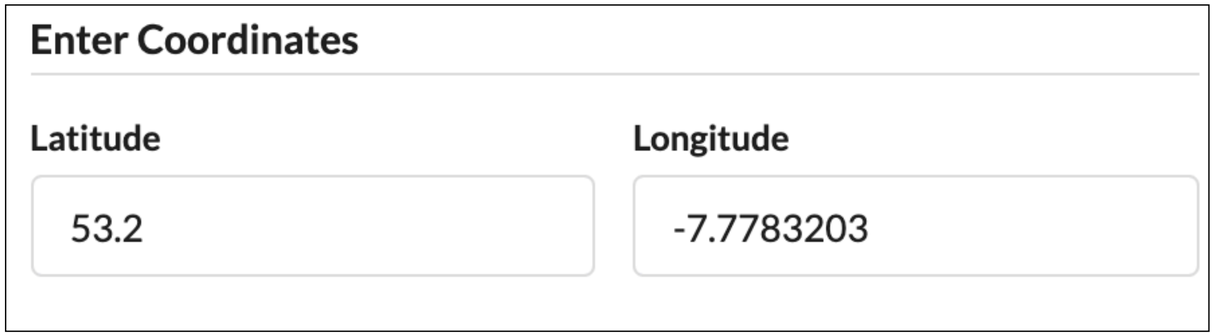
- Expand donation object to include location (lat/lng)



- Introduce new Location type
- Include as part of Donation type

src/services/donation-types.ts

```ts
export interface Location {
  lat: number;
  lng: number;
}

export interface Donation {
  amount: number;
  method: string;
  candidate: Candidate;
  location: Location;
}
```



coordinates Custom Element

src/resources/elements/coordinates.html

```
<template>
  <h4 class="ui dividing header"> Enter Coordinates </h4>
  <div class="two fields">
    <div class="field">
      <label>Latitude</label> <input placeholder="00.000000" value.bind="location.lat"/>
    </div>
    <div class="field">
      <label>Longitude</label> <input placeholder="00.000000" value.bind="location.lng"/>
    </div>
  </div>
</template>
```

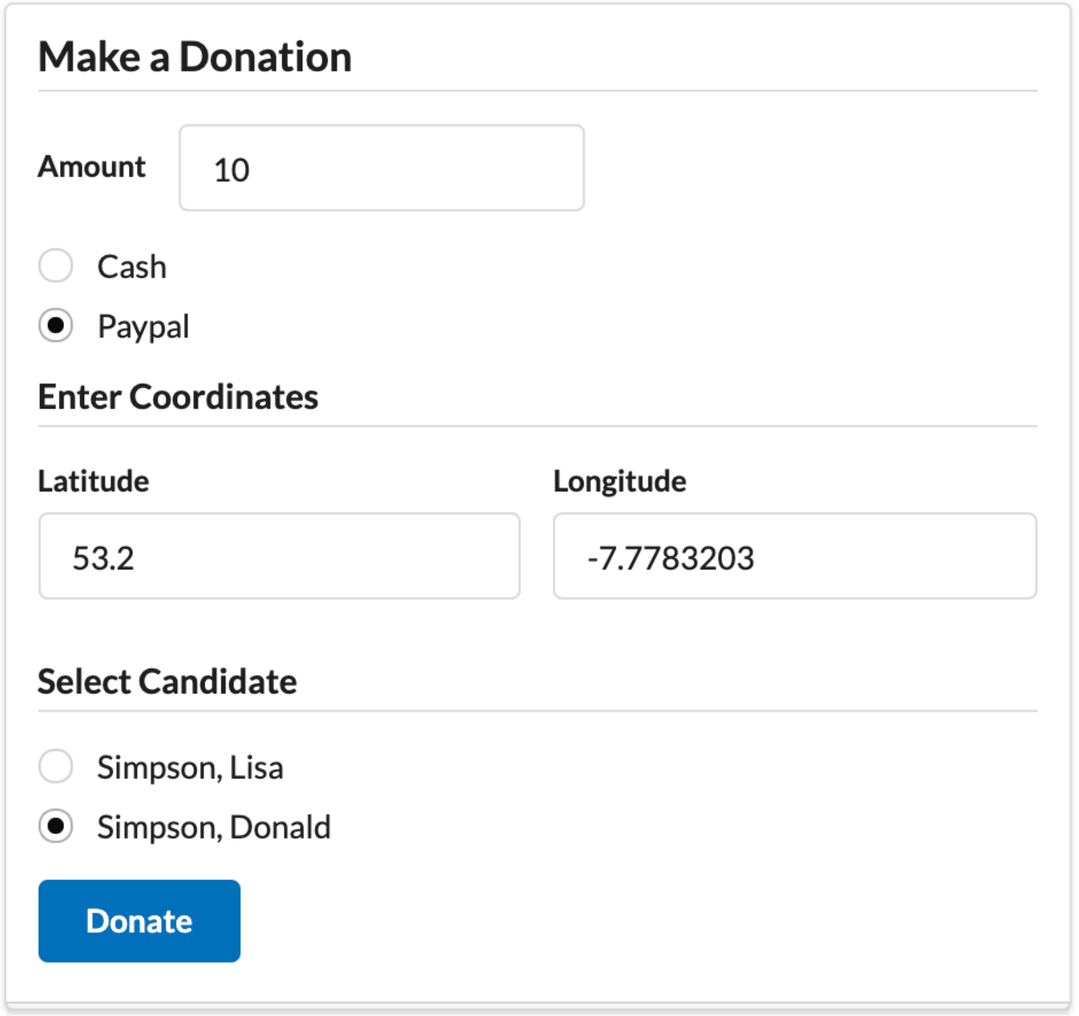
src/resources/elements/coordinates.ts

```
import { bindable } from 'aurelia-framework';
import { Location } from '../../services/donation-types';

export class Coordinates {
  @bindable location: Location;
}
```

```
        <label>${method}</label>
      </div>
    </div>
  </div>
  <coordinates location.two-way="location"></coordinates>
  <h4 class="ui dividing header"> Select Candidate </h4>
  <div class="grouped inline fields">
```

DonateForm

```
@inject(DonationService)
export class DonateForm {
  @bindable paymentMethods: string[];
  @bindable candidates: Candidate[];

  amount = '0';
  selectedMethod = '';
  selectedCandidate: Candidate = null;

  location: Location = { lat: 53.2734, lng: -7.7783203 };

  constructor(private ds: DonationService) {}

  makeDonation() {
    this.ds.donate(parseInt(this.amount), this.selectedMethod, this.selectedCandidate, this.location);
  }
}
```



DonationService

```
async donate(amount: number, method: string, candidate: Candidate, location : Location) {
  const donation = {
    amount: amount,
    method: method,
    candidate: candidate,
    location : location
  };
  const response = await this.httpClient.post('/api/candidates/' + candidate._id + '/donations', donation);
  this.donations.push(donation);
  this.total = this.total + amount;
  this.ea.publish(new TotalUpdate(this.total));
  console.log('Total so far ' + this.total);
}
```

Make a Donation

Amount 10

- ○ Cash
- ◉ Paypal

Enter Coordinates

Latitude 53.2

Longitude -7.7783203

Select Candidate

- ○ Simpson, Lisa
- ◉ Simpson, Donald

Donate

15
DONATED

Donations to Date

Amount	Payment Method	Candidate
5	Cash	Simpson, Lisa
10	Paypal	Simpson, Donald

Donate | Candidate | Logout

Make a Donation

Amount: 6

- (•) Cash
- ( ) Paypal

Enter Coordinates

Latitude: 53.2734

Longitude: -7.77832(

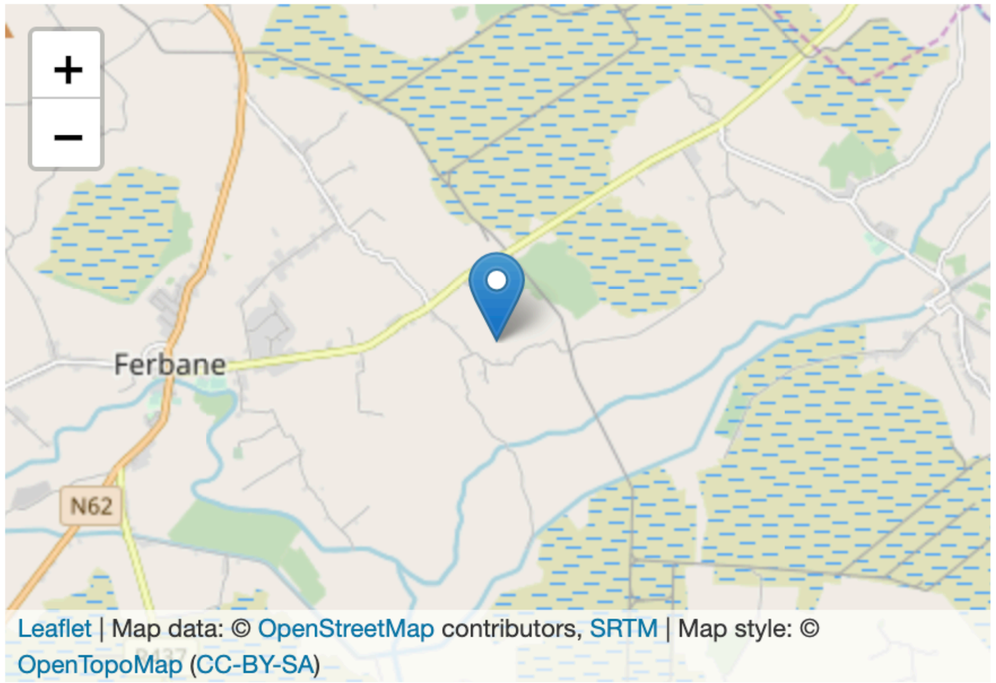
Select Candidate

- (•) Simpson, Lisa
- ( ) Simpson, Donald

Donate

6
DONATED

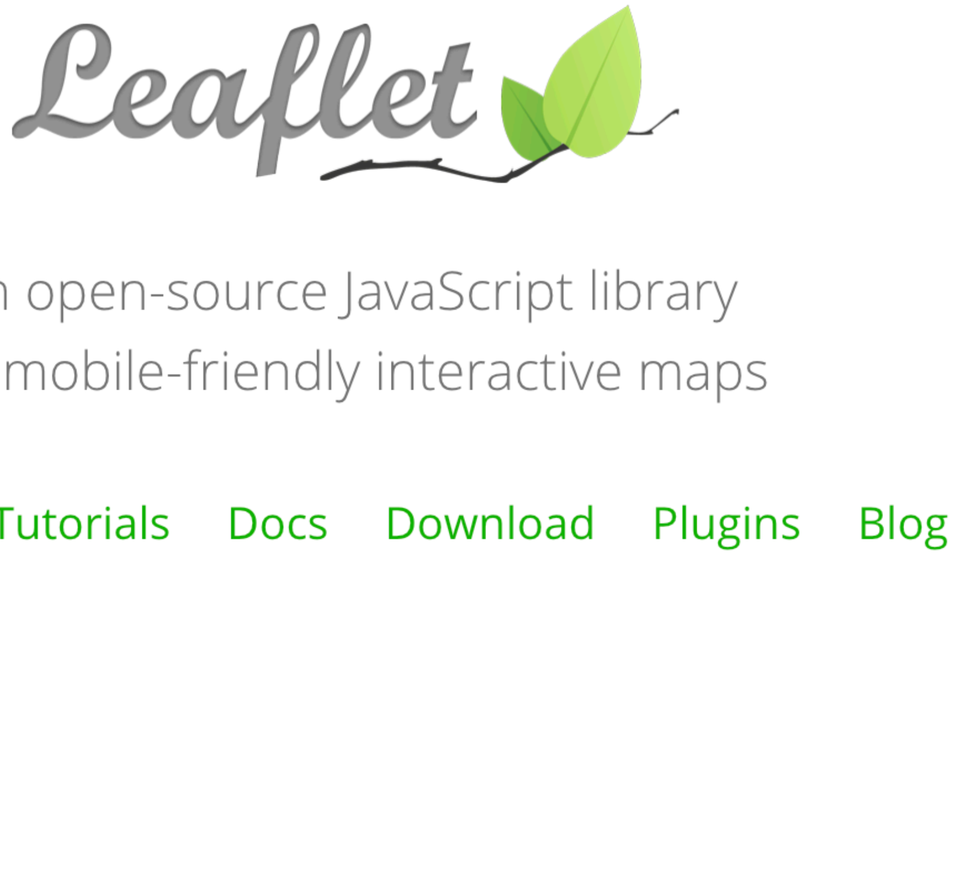
Introduce Map Component to Display Marker on location of donation



Donations to Date

Amount	Payment Method	Candidate
6	Cash	Simpson, Lisa

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

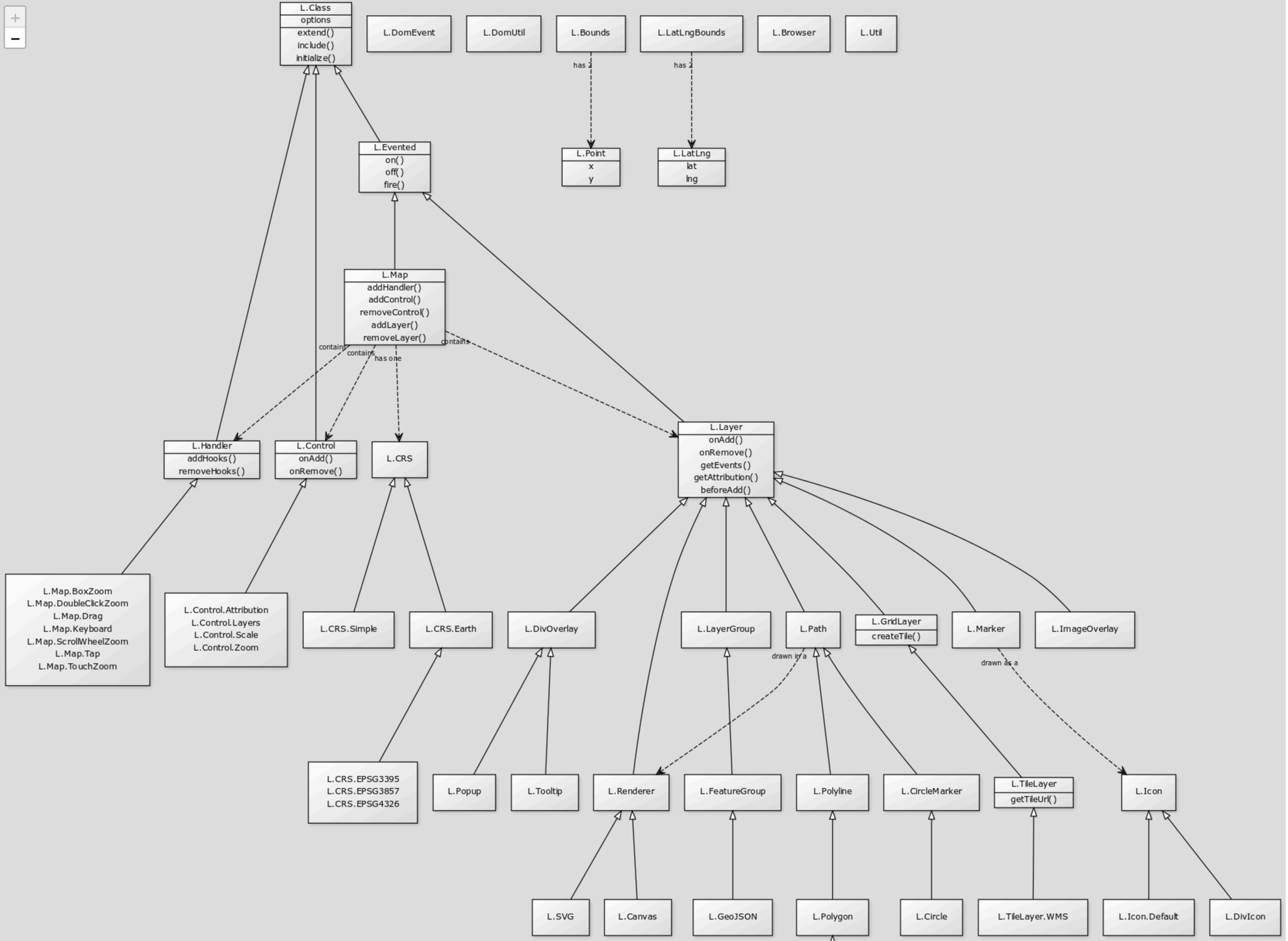
DomEvent
DomUtil
PosAnimation
Draggable

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 Aurelia

- Download and make available the leaflet javascript library to the Aurelia components

```
yarn add leaflet
```

- Include Leaflet style library in main application styles

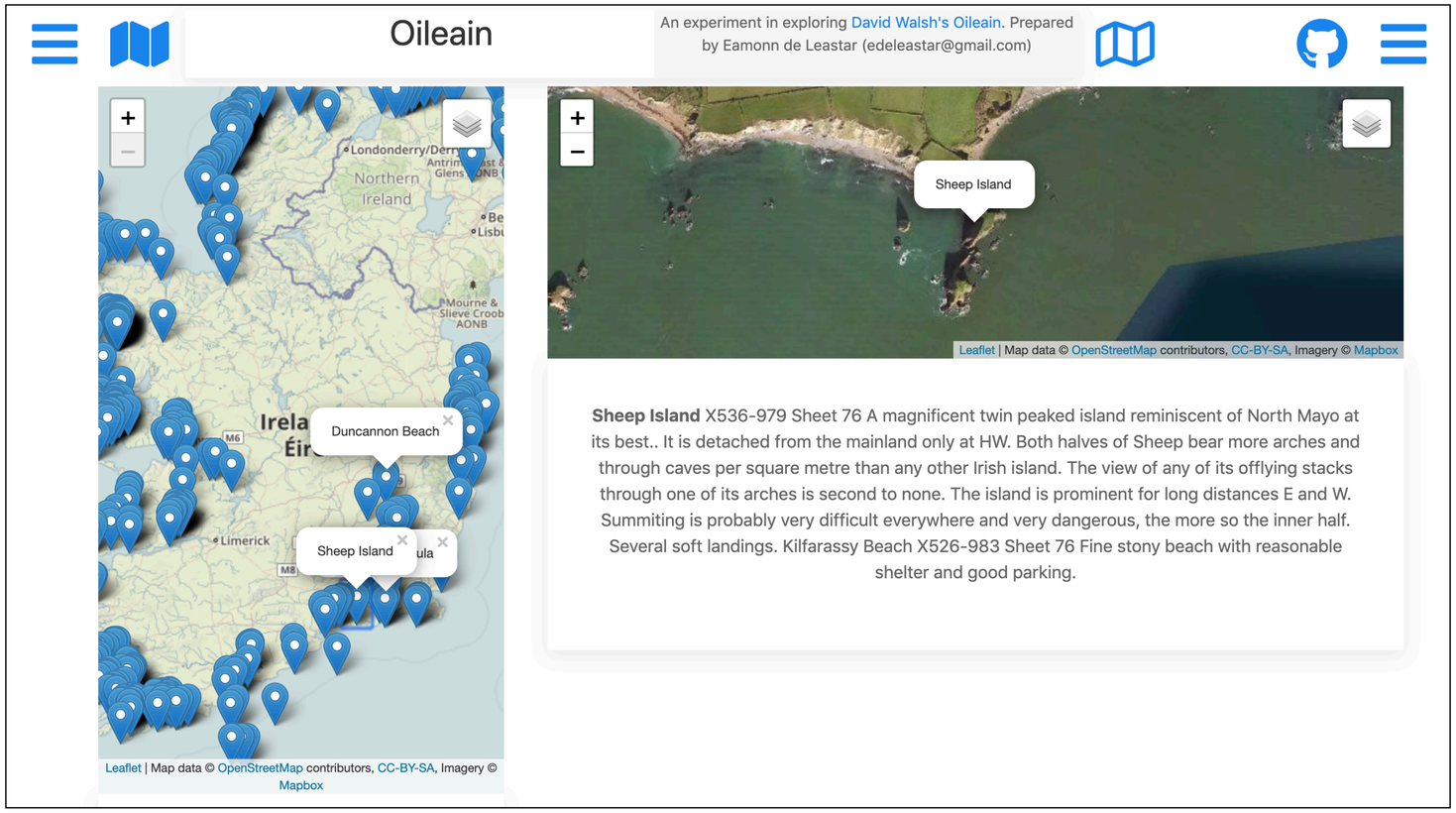
index.ejs

```
...
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.4.0/leaflet.css">
...
```

Wrapper Class - LeafletMap

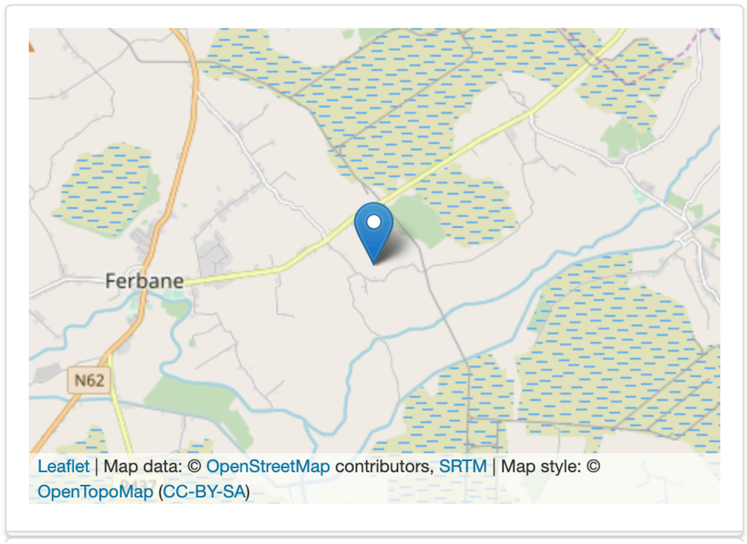
- Encapsulation of Leaflet to simplify map, maker, popup and layer management

- Developed to simplify Oileain web app development




```typescript
import * as L from 'leaflet';
import Map = L.Map;
import Layer = L.Layer;
import LayersObject = L.Control.LayersObject;
import LayerGroup = L.LayerGroup;
import LayerControl = L.Control.Layers;
import { Location } from './donation-types';

export interface MapConfig {
  location: Location;
  zoom: number;
  minZoom: number;
}

export class LeafletMap {
  imap: Map;
  control: LayerControl;
  overlays: LayersObject = {};

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

  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
    })
  };

  constructor(id: string, descriptor: MapConfig, activeLayer = '') {
    let defaultLayer = this.baseLayers.Terrain;
    if (activeLayer) {
      defaultLayer = this.baseLayers[activeLayer];
    }
    this.imap = L.map(id, {
      center: [descriptor.location.lat, descriptor.location.lng],
      zoom: descriptor.zoom,
      minZoom: descriptor.minZoom,
      zoomControl: false,
      layers: [defaultLayer]
    });
  }

  addLayer(title: string, layer: Layer) {
    this.overlays[title] = layer;
    this.imap.addLayer(layer);
  }

  addLayerGroup(title: string) {
    this.overlays[title] = L.layerGroup([]);
    this.imap.addLayer(this.overlays[title]);
  }

  showLayerControl() {
    this.control = L.control.layers(this.baseLayers, this.overlays).addTo(this.imap);
  }

  showZoomControl(position = 'topleft') {
    L.control
      .zoom({
        position: position
      })
      .addTo(this.imap);
  }

  moveTo(zoom: number, location: Location) {
    this.imap.setZoom(zoom);
    this.imap.panTo(new L.LatLng(location.lat, location.lng));
  }

  zoomTo(location: Location) {
    this.imap.setView(new L.LatLng(location.lat, location.lng), 8);
  }

  addMarker(location: Location, popupText = '', layerTitle = 'default') {
    let group: LayerGroup;
    let marker = L.marker([location.lat, location.lng]);
    if (popupText) {
      var popup = L.popup({ autoClose: false, closeOnClick: false });
      popup.setContent(popupText);
      marker.bindPopup(popup);
    }
    if (!this.overlays[layerTitle]) {
      group = L.layerGroup([]);
      this.overlays[layerTitle] = group;
      this.imap.addLayer(group);
    } else {
      group = this.overlays[layerTitle] as LayerGroup;
    }
    marker.addTo(group);
  }

  invalidateSize() {
    this.imap.invalidateSize();
    let hiddenMethodMap = this.imap as any;
    hiddenMethodMap._onResize();
  }
}
```

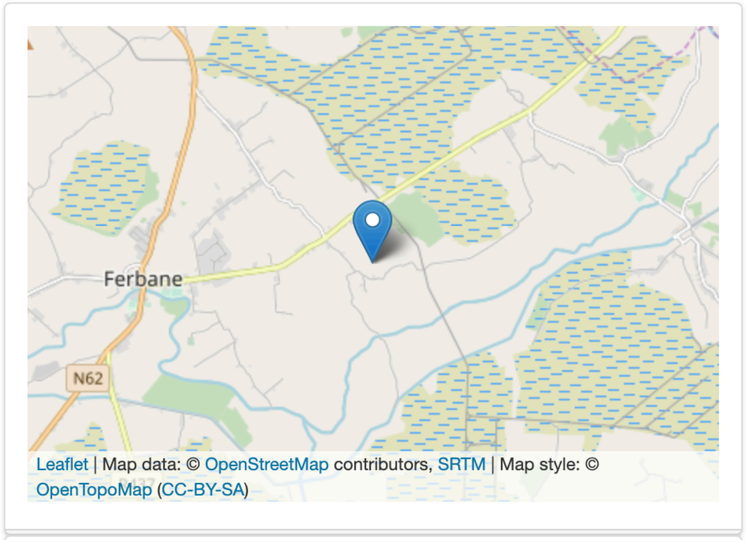
DonationMap Custom Element




```
<template>
  <div class="ui stacked segment">
    <div id="${mapId}" class="ui embed" style="height:${mapHeight}px; z-index: 0"></div>
  </div>
</template>
```

```
export class DonationMap {
  mapId = 'donations-map';
  mapHeight = 300;
  map: LeafletMap;

  constructor(private ea: EventAggregator) {}

  attached() {
    const mapConfig = {
      location: { lat: 53.2734, lng: -7.7783203 },
      zoom: 8,
      minZoom: 7
    };
    this.map = new LeafletMap(this.mapId, mapConfig, 'Terrain');
  }
}
```

DonationMap Custom Element



Element ID for Leaflet Map

Height for leaflet map

```
<template>
  <div class="ui stacked segment">
    <div id="${mapId}" class="ui embed" style="height:${mapHeight}px; z-index: 0"></div>
  </div>
</template>
```

Both of these defined in the ViewModel class

Unique ID
Map Height in pixels
Map object reference

Map Center + zoom settings and limit

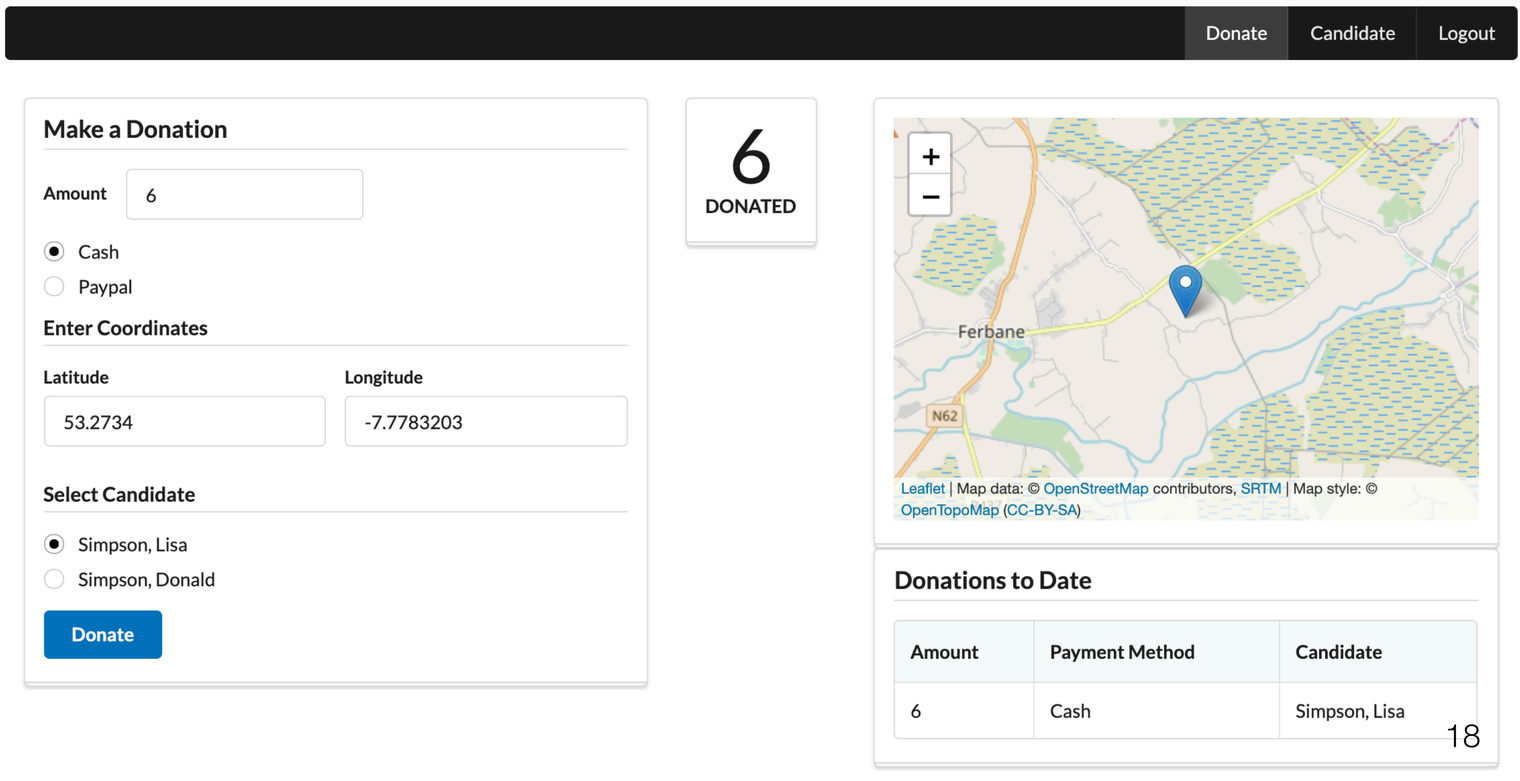
Create the map

```
export class DonationMap {
  mapId = 'donations-map';
  mapHeight = 300;
  map: LeafletMap;

  constructor(private ea: EventAggregator) {}

  attached() {
    const mapConfig = {
      location: { lat: 53.2734, lng: -7.7783203 },
      zoom: 8,
      minZoom: 7
    };
    this.map = new LeafletMap(this.mapId, mapConfig, 'Terrain');
  }
}
```

Default Layer

```
<template>
  <require from="../resources/elements/donations-list"></require>
  <require from="../resources/elements/donate-form"></require>
  <require from="../resources/elements/total-donated"></require>
  <require from="../resources/elements/donation-map"></require>

  <div class="ui stackable grid">
    <div class="seven wide column">
      <donate-form payment-methods.bind="paymentMethods" candidates.bind="candidates"></donate-form>
    </div>
    <div class="two wide column">
      <total-donated></total-donated>
    </div>
    <div class="seven wide column">
      <donation-map></donation-map>
      <donations-list donations.bind="donations"></donations-list>
    </div>
  </div>
</template>
```

- Include <donation-map> component



Donation Events

```
export class TotalUpdate {
  total: number;
  constructor(total: number) {
    this.total = total;
  }
}
```

```
import {Donation} from "./donation-types";

export class TotalUpdate {
  total: number;
  donation : Donation;
  constructor(total: number, donation: Donation) {
    this.total = total;
    this.donation = donation;
  }
}
```

- TotalUpdate events broadcast latest total value

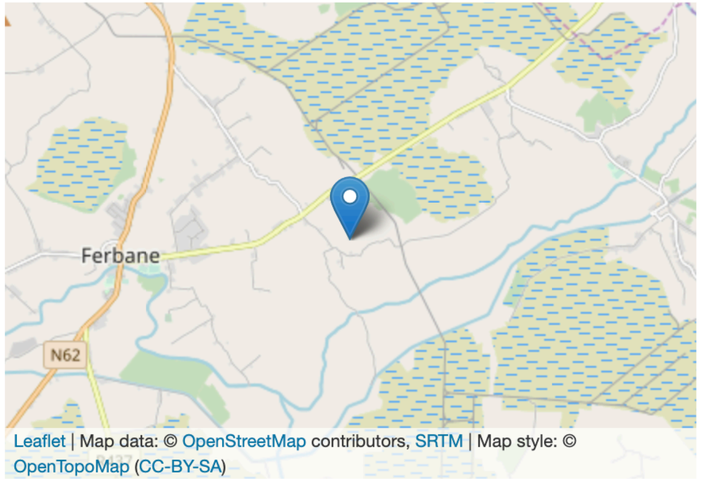
- Augment TotalUpdate to include latest donation object

```
async donate(amount: number, method: string, candidate: Candidate, location : Location) {
  const donation = {
    amount: amount,
    method: method,
    candidate: candidate,
    location : location
  };
  const response = await this.httpClient.post('/api/candidates/' + candidate._id + '/donations', donation);
  this.donations.push(donation);
  this.total = this.total + amount;
  this.ea.publish(new TotalUpdate(this.total, donation));
  console.log('Total so far ' + this.total);
}
```

- Broadcast TotalUpdate event containing donation to all interested subscribers

Subscribe to TotalUpdate events

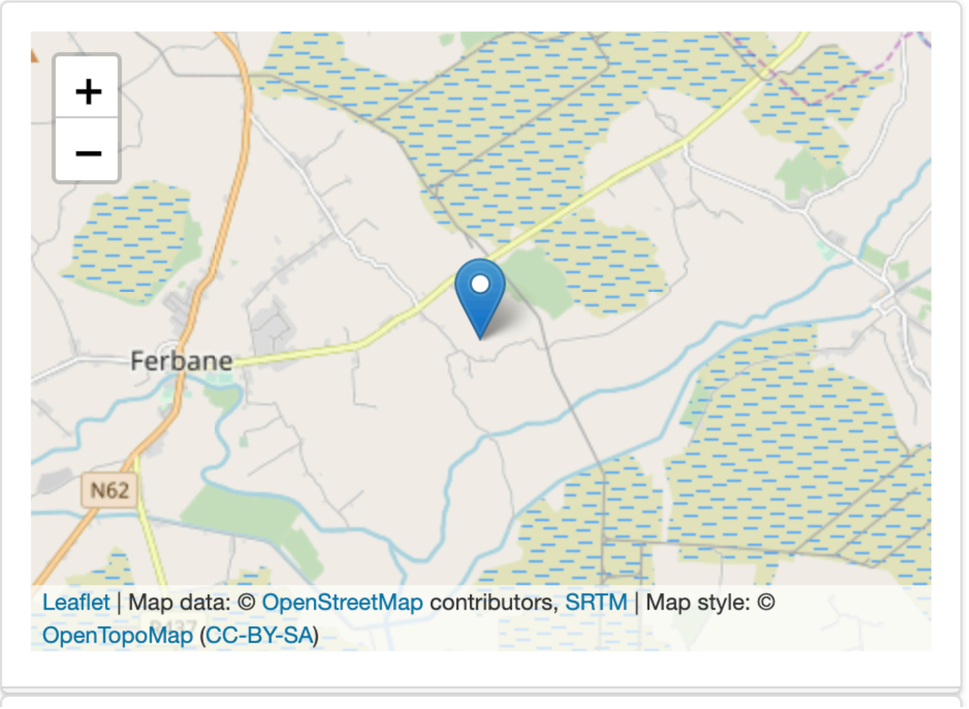
Add a new marker - and reset viewport to centre on new marker




```
@inject(EventAggregator)
export class DonationMap {
  mapId = 'donations-map';
  mapHeight = 300;
  map: LeafletMap;

  constructor(private ea: EventAggregator) {
    ea.subscribe(TotalUpdate, msg => {
      this.renderDonation(msg.donation);
    });
  }

  renderDonation(donation: Donation) {
    if (this.map) {
      this.map.addMarker(donation.location);
      this.map.moveTo(12, donation.location);
    }
  }

  attached() {
    const mapConfig = {
      location: { lat: 53.2734, lng: -7.7783203 },
      zoom: 8,
      minZoom: 7
    };
    this.map = new LeafletMap(this.mapId, mapConfig, 'Terrain');
  }
}
```

Zoom Control

```
  attached() {
  const mapConfig = {
    location: { lat: 53.2734, lng: -7.7783203 },
    zoom: 8,
    minZoom: 7
  };
  this.map = new LeafletMap(this.mapId, mapConfig, 'Terrain');
  this.map.showZoomControl();
}
```



- Zoom Control will be limited by minZoom constraint

```
const donationStr = `${donation.candidate.firstName} ${donation.candidate.lastName} €${donation.amount.toString()}`;
this.map.addMarker(donation.location, donationStr);
```

- Associate popup string with marker when adding
- Popup displayed when marker clicked
- Must be explicitly closed