Leaflet Tutorial

An open-source JavaScript library
for mobile-friendly interactive maps

Pierre-André Le Ny
peter.leny@obscom.eu

Preamble

In this tutorial, we will use basic HTML, CSS, and JavaScript (specifically, the Leaflet JavaScript library) to create a series of progressively complex web maps.

- **HTML** (HyperText Markup Language) allows us to structure content for our web page. Our map will be contained in an element within an HTML file.

- **CSS** (Cascading Style Sheets) gives us control of the style and visual presentation of our web page. We will use it in the placement and sizing of the map and to customize some Leaflet elements.

- **JS** (JavaScript) gives us the ability to add interactivity to our web page. We will use it to pull in map tiles to our web page, add data (or content layers), and handle user interaction with the map.

What is Leaflet?

Leaflet is an open-source JavaScript library that gives us code to create interactive, mobile friendly web maps. Think of it as a collection (or library) of prewritten JavaScript that does some of the heavy lifting/scripting of web map stuff for us. We will interact with the library through its well documented API. It has a small file size but is packed with useful features and can be extended even further with plugins.

Table of contents

1. Setup

1.1. Environement

For this tutorial you will need:

- a computer
- internet access
- a web browser (most any modern browser)
- a text or code editor (notepad, textedit, sublime text, atom editor, etc)

In case of error in your code, you must use the developer tools extension in your browser.

1.2. Links

- Official website:

  http://leafletjs.com/

- Source-code:

  https://github.com/Leaflet/Leaflet

- Video of Leaflet creator - Vladimir Agafonkin - How Simplicity Will Save GIS:

  http://vimeo.com/106112939

- JavaScript Guide:

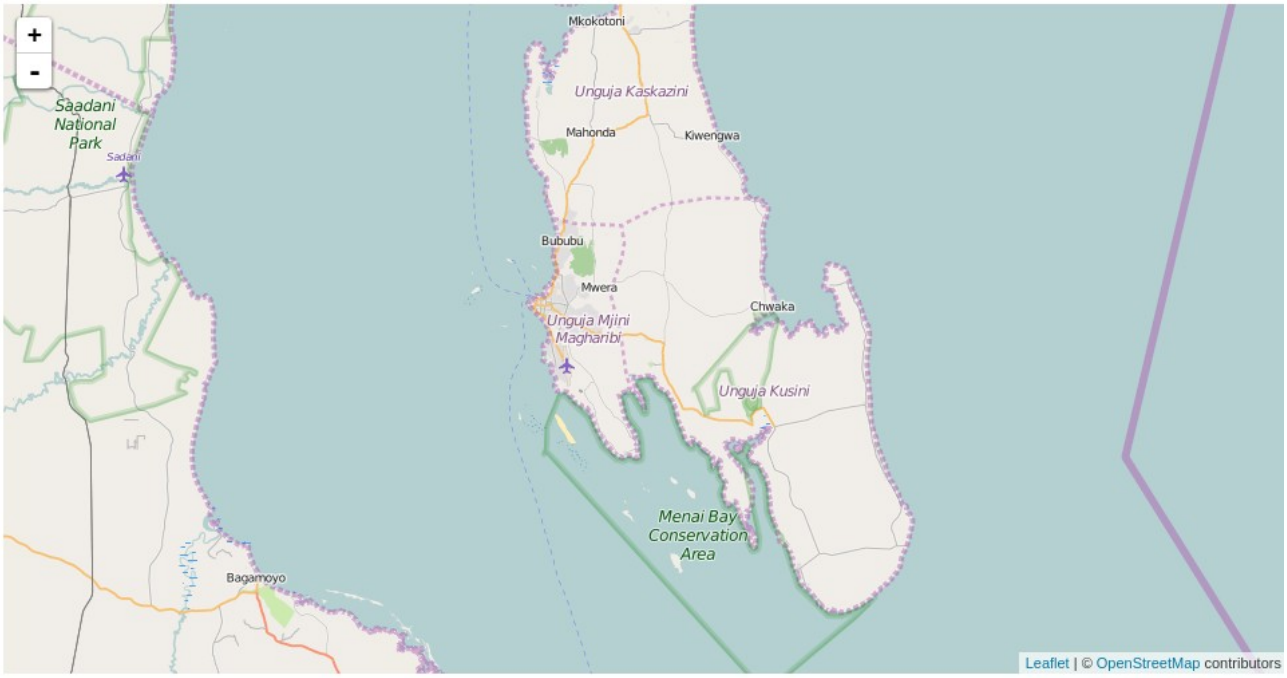
  https://developer.mozilla.org/en/docs/JavaScript/Guide

- Test your JavaScript code:

  http://jsfiddle.net/

2. Basic Map of Zanzibar

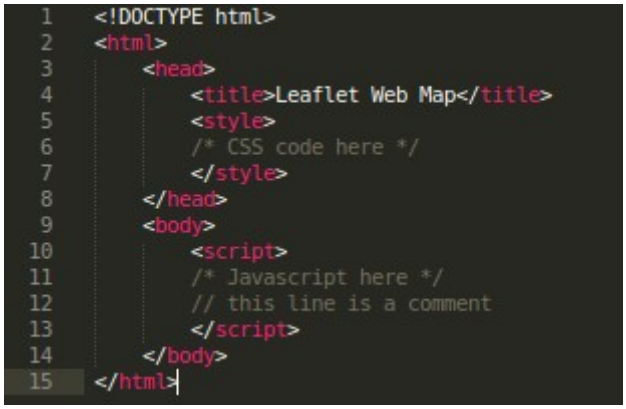
Here is the base map centered on Zanzibar we will build first:



2.1. Create a basic HTML document

Open a file editor > Create a new file > Save the file using a .html extension.

Next, set up the structure of the web page by adding the following markup to the file:



Reference the Leaflet CSS and JavaScript Files

To make use of Leaflet we need to reference its CSS and JavaScript files in the HTML file.

You can reference the hosted Leaflet files or download copies, host them locally, and reference those. Let's reference the hosted files. To do so, insert the following code into the <head> section of the HTML page:

```
<link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet/v0.7.7/leaflet.css" />

<script src="http://cdn.leafletjs.com/leaflet/v0.7.7/leaflet.js"></script>
```

CDN means Content Delivery Network. In this cas, the library is online.

Add Map Div and Style It

Leaflet requires a element to contain the map and that that element have a set height.

Create a element with the id of "map" to contain the map by adding the following to the HTML <body>:

```
<div id="map"></div>
```

To create a map that is 960 px by 500 px add the following CSS code between the <style> tags in the <head> section of the HTML page:

```
#map {
```

```
width: 960px;
height: 500px;
}
```

Now the HTML and CSS are set. The basic structure and style of our web page is in place. All we need to do is add some JavaScript and we'll have a web map!

Initialize Map

The first script we will write pulls in some map tiles and configures a few basic map settings.

Enter the following JavaScript between the <script> tags of the HTML file.

```
<script>

    var map = L.map('map',{
    center: [-6.19922, 39.30686],
    zoom: 10
    });

    L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
    }).addTo(map);

</script>
```

-6.19922, 39.30686 are the coordinates (lat-long) of the Tunguu campus on WGS84 projection.

The code is now complete so the code should now look like this:

```
<!DOCTYPE html>
<html>
<head>
<title>Leaflet Web Map</title>
<link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet/v0.7.7/leaflet.css" />
<script src="http://cdn.leafletjs.com/leaflet/v0.7.7/leaflet.js"></script>
<style>
#map {
   width: 960px;
   height:500px;
}
</style>
</head>

<body>
   <div id="map"></div>
<script>
   var map = L.map('map',{
   center: [-6.19922, 39.30686],
   zoom: 15
   });

   L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
   attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
   }).addTo(map);

</script>
</body>
</html>
```

Here is breakdown of the script:

The script creates the 'map' variable, assigns it a new L.map object and passes it the id (‘map’ in this case) of the div element in which the map is to be contained. The script goes on to pass some options that set an initial center point and zoom level for the map. Essentially, this creates a map on the page that we can manipulate.

Next, the script creates a new L.tileLayer object, specifying a particular set of tiles to be loaded into the map container and passes in an 'attribution' option. In this case OpenStreetMap tiles are used but there are many map tile providers. Experiment with different sets and remember to always properly attribute the data and imagery used! As tiles are the basis of web maps, it is worth learning more about them. For detailed information see Anatomy of a Web Map, by Alan McConchie and Beth Schechter of Maptime and Stamen.

Finally, the addTo() method is used to add the tile layer to the map.

Make sure to save the file and then open it with a web browser.

You should see a web map centered on the Tunguu offices!

2.2. Modify the Base Map

Let's go ahead and make two minor adjustments to the script. The map will look the same but be a

little more user friendly. The code will be sleeker as well.

```
<script>
   var map = L.map('map',{scrollWheelZoom:false}).setView([-6.19922, 39.30686], 15);

   L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
   attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
   }).addTo(map);

</script>
```

First, remove the 'center' and 'zoom' options from the L.map constructor.

Next, modify the script to deactivate the scroll WheelZoom interactivity option so that we don't accidentally zoom the map when we are trying to scroll down the page. This is done by passing 'scrollWheelZoom: false' as an option to L.map(). This can be a particularly important usability issue if the map is large or if there is content above and/or below it.

Finally, set the initial center and zoom level with the setView() method.

The map should look just like our first one but we can't control the zoom level by scrolling and the code is a bit streamlined.

Now we have a customized base map with which to build on.

2.3. Add a WMS Layer to the Map

This example adds a Web Map Service (WMS) Overlay to the map.

```
<script>

   var map = L.map('map').setView([-6.19922, 39.30686], 3);

   L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
   attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
   }).addTo(map);

   L.tileLayer.wms("http://195.154.41.198/geoserver/geonode/wms?service=WMS&version=1.1.0", {
         layers: 'geonode:kibele_merged_v2_transparent_mosaic_group1',
      format: 'image/png',
      transparent: true,
      attribution: "Suza - World Bak Drone Imagery"
   }).addTo(map);

</script>
```

This code changes the zoom level in the setView() method to 3 and using L.tileLayer.wms() displays a WMS service as a tile layer on the map. Like the L.tilelayer() function, L.tilelayer.wms() is passed the url to the service provider and a set of options.

2.4. Add a Default Marker to the Map

This example adds a default Leaflet marker on the Tunguu offices.

```
<script>
   var marker = L.marker([-6.19922, 39.30686]).addTo(map);
</script>
```

This code creates a marker variable and assigns it a marker object with a particular point using L.marker(). The marker is then added to the map with the addTo() method.

2.5. Add a Custom Marker to the Map

This example adds a custom marker (a customized icon) on the Tunguu offices.

Choose a SVG icon http://www.flaticon.com/ , download it and put it in a folder called img.

```
<script>

   var map = L.map('map', {scrollWheelZoom:false}).setView([-6.19922, 39.30686], 15);

   L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
   attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
   }).addTo(map);

   var MyIcon = L.icon({
      iconUrl: 'http://localhost/…/urltheicon.svg',
      iconSize: [38, 95], // size of the icon
      });

   var marker = L.marker([-6.19922, 39.30686], {icon: MyIcon}).addTo(map);

</script>
```

This code creates the ‘MyIcon’ variable and assigns it an icon marker using L.icon().

A few options are passed to L.icon():

iconUrl: the url to an image to be used as icon marker
iconSize: thesize to make the icon marker

The ‘MyIcon’ variable is then passed to the ‘icon’ option of L.marker().

More information : http://leafletjs.com/examples/custom-icons.html

2.6. Add a Popup to the Marker

This example adds a standard popup to the custom marker. It is activated upon click.

```
<script>

   var map = L.map('map', {scrollWheelZoom:false}).setView([-6.19922, 39.30686], 15);

   L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
   attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
   }).addTo(map);

   var MyIcon = L.icon({
      iconUrl: 'url_to_your_icon',
      iconSize: [38, 95], // size of the icon
      });

   L.marker([-6.19922, 39.30686], {icon: MyIcon}).bindPopup('Tunguu Office').addTo(map);

</script>
```

This code adds a popup to marker click by using the bindPopup() method.

2.7. Customize the Popup on the Marker

This example adds HTML content and a custom style to the popup.

```
<style>
#map {
   width: 960px;
   height:500px;
}
.custom .leaflet-popup-tip,
.custom .leaflet-popup-content-wrapper {
   background: #e93434;
   color: #ffffff;
}
</style>
```

This CSS code is added to the head of the HTML file. It adds styles for the class 'custom' which we will create in our script.

```
<script>

   var map = L.map('map', {scrollWheelZoom:false}).setView([-6.19922, 39.30686], 15);

   L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
   attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
   }).addTo(map);

   var MyIcon = L.icon({
      iconUrl: 'URLicon',
      iconSize: [38, 95], // size of the icon
      popupAnchor: [0,-15]
      });
```

```
    var customPopup = "SUZA University<br/><img src='logo.gif' alt='logo gif' width='350px'/>";

    var customOptions =
        {
        'maxWidth': '500',
        'className' : 'custom'
        }

    L.marker([-6.19922, 39.30686], {icon: MyIcon}).bindPopup(customPopup,customOptions).addTo(map);

</script>
```

This code first adds a 'popupAnchor' option to the icon. The values set moves the popup higher so that more of the icon is visible.

Then it creates the variable ‘customPopup’ and assigns it some HTML content.

Next it creates the variable ‘customOptions’ and specifies a few options:

-the class name 'custom' is added so custom CSS styles can be applied
-a maxWidth is set*

The bindPopup method takes HTML content and options as arguments. The variables created are passed into the method.

*Note, by default the maximum width of a popup is set to 300px. Sometimes you might want it wider, particularly if you want to add content to it that is larger than 300px. To avoid content spilling out of the popup, make sure the 'maxWidth' option of popup is set larger than the width of content used.

2.8. Mobile ready

Put this CSS code in the style section :

```
body { padding: 0; margin: 0; }

html, body, #map { height: 100%; }
```

Add this meta tag at the top of your page to enable a mobile vizualisation :

```
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no" />
```

2.9. Geolocation

Leaflet has a very handy shortcut for zooming the map view to the detected location — locate method with the setView option, replacing the usual setView method in the code:

```
map.locate({setView: true, maxZoom: 16});
```

Here we specify 16 as the maximum zoom when setting the map view automatically. As soon as the user agrees to share its location and it's detected by the browser, the map will set the view to it. Now we have a working fullscreen mobile map!

But what if we need to do something after the geolocation completed? Here's what the locationfound and locationerror events are for. Let's for example add a marker in the detected location, showing accuracy in a popup, by adding an event listener to locationfound event before the locateAndSetViewcall:

```
function onLocationFound(e) {
    var radius = e.accuracy / 2;
    L.marker(e.latlng).addTo(map)
        .bindPopup("You are within " + radius + " meters from this point").openPopup();
    L.circle(e.latlng, radius).addTo(map);
}

map.on('locationfound', onLocationFound);
```

Excellent! But it would also be nice to show an error message if the geolocation failed:

```
function onLocationError(e) {
    alert(e.message);
}
map.on('locationerror', onLocationError);
```

If you have setView option set to true and the geolocation failed, it will set the view to the whole world.

2.10. Add a Geojson file

Keep in mind that QGIS, desktop GIS software, can export your shapefiles directly in Geojson. Projection (SRC) must be EPSG:4326 (WGS84).

There are many possibilities to display a Geojson with Leaflet.

http://leafletjs.com/examples/geojson.html

https://github.com/calvinmetcalf/leaflet-ajax

http://savaslabs.com/2015/05/18/mapping-geojson.html

Create a new folder called data and put it the file hotels.json.

```
<script src="data/hotels.js" type="text/javascript"></script>
```

Add this code to your main index.html file :

```
//Add Hotels GeoJSON
        var Hotels_layer = L.geoJson(hotels, { style: function (feature) });
        Hotels_layer .addTo(map);
```

2.11. Add a scalebar

Scalebar is a native control of Leaflet. It is very simple to add it to your map.

```
//scalebar
L.control.scale().addTo(map);
```

2.12. Titles and subtitles

We are going to use a new control to change our page in order to ddd some new titles and subtitles in some div components.

```
var title = new L.Control();
		title.onAdd = function (map) {
			this._div = L.DomUtil.create('div', 'info'); // create a div with a class "info"
			this.update();
			return this._div;
	};
	title.update = function () {
			this._div.innerHTML = '<h2>This is the title</h2>This is the subtitle'
		};
		title.addTo(map);
```

and the CSS code :

```
.info {
    padding: 6px 8px;
    font: 14px/16px Arial, Helvetica, sans-serif;
    background: white;
    background: rgba(255,255,255,0.8);
    box-shadow: 0 0 15px rgba(0,0,0,0.2);
    border-radius: 5px;
        }
    .info h2 {
    margin: 0 0 5px;
    color: #777;
}
```

Modify the text of the layer by this one :

```
this._div.innerHTML = '<center><h2>Zanzibar Interactive Map</h2>2016<br><img src="img/logo_suza.png"
/><br></center>'
```

2.13. Define a new baselayer

At this time, you will be able to add a new baselayer instead of OSM Layer.

Stamen Design delivers some nice base layers by using a specific tile format.

http://tile.stamen.com/toner/{z}/{x}/{y}.png

Leaflet displays a map by displaying a series of tiles across the page. It needs to know where to get these tiles from and how the tiles are stored in a directory structure.

- Tiles are 256 × 256 pixel PNG files
- Each zoom level is a directory, each column is a subdirectory, and each tile in that column is a file
- Filename(url) format is /zoom/x/y.png

The slippy map expects tiles to be served up at URLs following this scheme, so all tile server URLs

look pretty similar.

All tiles are 256 x 256 pixel PNG files, layed out on a square grid, numbered starting with 0 at the top left of the globe. At the lowest zoom level, zoom 0, one tile covers the entire world. The tiles are named as {z}/{x}/{y}.png, where z is zoom, x is the tile number from left to right, and y is the tile number from top to bottom. For example, the solitary tile on zoom 0 is named 0/0/0.png and looks like this:

First, comment the TileLayer OpenStreetMap by using /* your code */.

```
/*L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
    }).addTo(map);*/
```

Add these lines instead :

```
L.tileLayer('http://tile.stamen.com/toner/{z}/{x}/{y}.png', {
    attribution: '&copy; <a href="http://maps.stamen.com/">Stamen Design</a>'
    }).addTo(map);
```

You will see on your map a black&white base layer.

Stamen provides other base layers :

http://tile.stamen.com/terrain/{z}/{x}/{y}.jpg

http://tile.stamen.com/watercolor/{z}/{x}/{y}.jpg

Feel free to use them.

Use your mapbox token or any other tilelayer (google, bing, https://leaflet-extras.github.io/leaflet-providers/preview/)

2.14. Layers Groups and layers controls

In this part, we are going to build a control to create a TOC (Table of Contents) in order to control to display or hide the layers.

First, we need to change the declaration of the base layers.

Add two new variables osm and stamen and remove the older declarations.

```
var osm = L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
});


var stamen = L.tileLayer('http://tile.stamen.com/toner/{z}/{x}/{y}.png', {
attribution: '&copy; <a href="http://maps.stamen.com">Stamen Design</a>'
});
```

Do the same for the kibele raster layer :

```
var kibele = L.tileLayer.wms("http://195.154.41.198/geoserver/geonode/wms?
service=WMS&version=1.1.0", {
    layers: 'geonode:kibele_merged_v2_transparent_mosaic_group1',
    format: 'image/png',
    transparent: true,
    attribution: "Suza - World Bak Drone Imagery"
});
```

Instead of using addTo(map) function, we are give some options to the var map declaration :

```
var map = L.map('map',{
center: [-6.19922, 39.30686],
zoom: 10,
layers: [stamen, kibele]
});
```

At the end of the main script tag, add these lines :

```
var baseLayers = {
"Stamen": stamen,
"OpenStreetMap": osm,
};
```

```
var overlays = {
"Raster" : kibele,
};
```

And finally, add the new TOC control to the map :

```
L.control.layers(baseLayers, overlays).addTo(map);
```

You must see a new control on the right-top of your webmap, materialized by this icon :

2.15. Using local library

Instead of using CDN online Leaflet library, we would like to use a local version.

Download the library here :

http://cdn.leafletjs.com/leaflet/v0.7.7/leaflet.zip

Extract the zip file in a temporary directory.

- Copy leaflet.js in your js directory
- Copy leaflet.css in your css directory
- Copy images directory in your wemap folder

Use local declaration in your code :

```
<script src="js/leaflet.js"></script>
```

Do the same thing for the CSS file.

2.16. Change the Geolocation function

Because, we don't want be located by default, we want to change the way of using the Geolocation function.

First, comment the function locate by adding // at the beginning of the line :

```
//map.locate({setView: true, maxZoom: 16});
```

We would like to create a new button (<input type=button>) to allow user to detect his location, instead of doing it by default.

Change the line :

```
this._div.innerHTML = '<center><h2>Zanzibar Interactive Map</h2>2016<br><img src="img/logo_suza.png" /><br></center>'
```

By this one :

```
this._div.innerHTML = '<h2>Zanzibar Interactive Map</h2>2016<br><center><img src="img/logo_suza.png" /><br><input type=button value="Locate me" onclick="map.locate({setView: true, maxZoom: 16})";/>'};
```

We are using the Onclick javascript native function in order to detect our localization.

We can add some CSS code to change the aspect of the button. We can use this online tool to generate the CSS code : http://button.csscook.com/

Add these CSS lines between your <style> tags :

```
.btn-style{
	border : solid 4px #e6e6e6;
	border-radius : 16px;
	moz-border-radius : 16px;
	-webkit-box-shadow : inset 0px 0px 2px rgba(0,0,0,1.0);
	-moz-box-shadow : inset 0px 0px 2px rgba(0,0,0,1.0);
	box-shadow : inset 0px 0px 2px rgba(0,0,0,1.0);
	font-size : 14px;
	color : #696869;
	padding : 9px 20px;
	background : #ffffff;
	background : -webkit-gradient(linear, left top, left bottom, color-stop(0%,#ffffff), color-stop(49%,#f1f1f1), color-stop(51%,#e1e1e1), color-stop(100%,#f6f6f6));
	background : -moz-linear-gradient(top, #ffffff 0%, #f1f1f1 49%, #e1e1e1 51%, #f6f6f6 100%);
	background : -webkit-linear-gradient(top, #ffffff 0%, #f1f1f1 49%, #e1e1e1 51%, #f6f6f6 100%);
	background : -o-linear-gradient(top, #ffffff 0%, #f1f1f1 49%, #e1e1e1 51%, #f6f6f6 100%);
	background : -ms-linear-gradient(top, #ffffff 0%, #f1f1f1 49%, #e1e1e1 51%, #f6f6f6 100%);
	background : linear-gradient(top, #ffffff 0%, #f1f1f1 49%, #e1e1e1 51%, #f6f6f6 100%);
	filter : progid:DXImageTransform.Microsoft.gradient( startColorstr='#ffffff', endColorstr='#f6f6f6',GradientType=0 );

}
```

Because in CSS, the dot specifies a class, you must add the code *class="btn-style"* to your input type=button declaration :

At the end, the code will be this one :

```
this._div.innerHTML = '<center><h2>Zanzibar Interactive Map</h2>2016<br><img src="img/logo_suza.png" /><br><input type=button class="btn-style" value="Locate me" onclick="map.locate({setView: true, maxZoom: 16})";/>';};
```

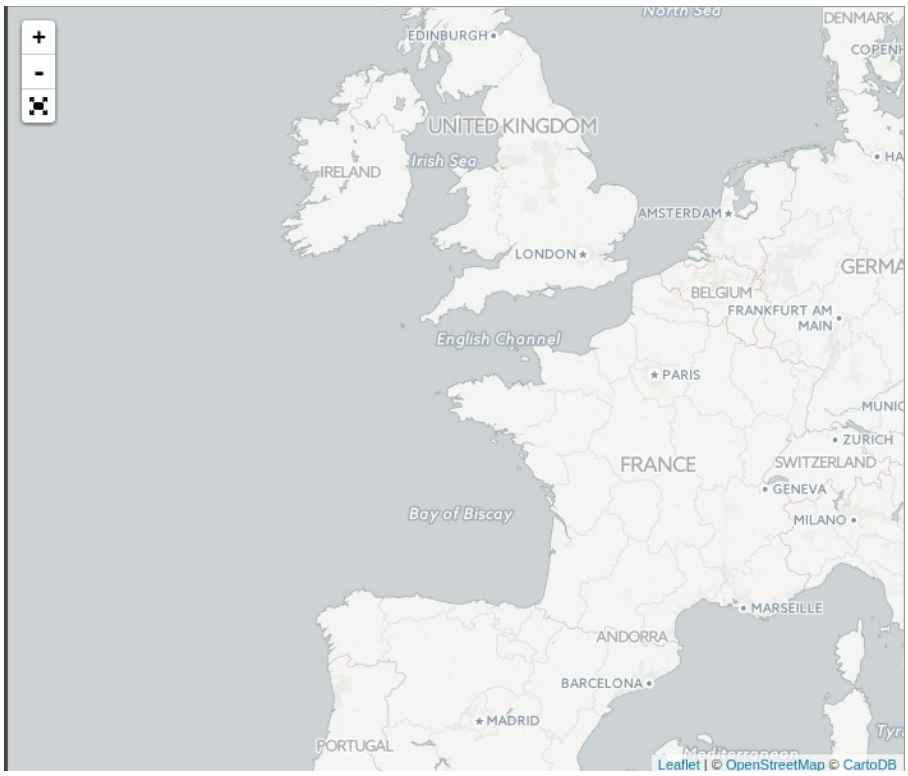
3. Plugins

3.1. FullScreen

Create a new folder called fullscreen.

Download the plugin here : http://brunob.github.io/leaflet.fullscreen/

Extract the zip in the fullscreen folder.



Include Control.FullScreen.js and Control.FullScreen.css in your page:

```
<link rel="stylesheet" href="fullscreen/Control.FullScreen.css" />
<script src="fullscreen/Control.FullScreen.js"></script>
```

Add the fullscreen control to the map by adding fullscreen options to the map (don't erase your previous options).

```
var map = new L.Map('map', {
  fullscreenControl: true,
  fullscreenControlOptions: {
    position: 'topleft'
  }
});
```

Be sure to call the fullscreen.js after the call of the main Leaflet library. L function must be known.

If your map have a zoomControl the fullscreen button will be added at the bottom of this one.

3.2. Shapefile

This part presents how to display a shapefile directly with Leaflet.

First, you have to download in your Zansea-Geonode Spatial Data Infrastructure (SDI) the Wards layer of Ugunja in WGS84 with a GetFeature WFS request.

http://195.154.41.198/geoserver/wfs?format_options=charset%3AUTF-8&typename=geonode%3Awards_wgs84&outputFormat=SHAPE-ZIP&version=1.0.0&service=WFS&request=GetFeature

With a plugin, it is possible to display a shapefile on your interactive map.

https://github.com/calvinmetcalf/leaflet.shapefile/archive/gh-pages.zip

Extract it in a new folder called shapefile at the root of your folder.

On your main code, add these lines :

```
<script src="shapefile/catiline.js"></script>
<script src="shapefile/leaflet.shpfile.js"></script>
```

Further in the code, add these lines :

```
var shpfile = new L.Shapefile('data/wards_wgs84.zip', {
        onEachFeature: function(feature, layer) {
            if (feature.properties) {
                layer.bindPopup(Object.keys(feature.properties).map(function(k) {
                    return k + ": " + feature.properties[k];
                }).join("<br />"), {
                    maxHeight: 200
                });
            }
        }
    });
        shpfile.once("data:loaded", function() {
        console.log("finished loaded shapefile");
    });
```

Be sure to copy first a zipped shapefile of wards in your data directory.
Add the shpfile in the overlays declaration :

```
var baseLayers = {
"Stamen": stamen,
"OpenStreetMap": osm,
};

var overlays = {
"Raster" : kibele,
"Meteo" : nexrad,
"Wards" : shpfile
};
```

On this example, we want to display a Meteo layer called nexrad. Here is the layer declaration :

```
var nexrad = L.tileLayer.wms("http://mesonet.agron.iastate.edu/cgi-bin/wms/nexrad/n0r.cgi", {
layers: 'nexrad-n0r-900913',
format: 'image/png',
transparent: true,
```

```
    attribution: "Weather data © 2012 IEM Nexrad"
});
```

The result must be visible on the map :

Note that with Chrome the layer will not be display if you are running the file locally. (i.e with a c:\users\…). No problem if you are using a server web to access to your page with http://localhost

There is no probleme with Firefox. Chrome is more strict about cross origin policy than Firefox is for local files.

Cross-origin resource sharing (CORS) is a mechanism that allows restricted resources (e.g. fonts) on a web page to be requested from another domain outside the domain from which the resource originated

We are going to change the style of the shapefile :

Add a new variable :

```
//style
    var myStyle = {
        "color": "#ff7800",
        "weight": 1,
        "opacity": 0.65
    };
```

And add the option *style: myStyle* to the shpfile var.

```
    var shpfile = new L.Shapefile('data/wards_wgs84.zip', {style: myStyle,
            onEachFeature: function(feature, layer) {
                if (feature.properties) {
                    layer.bindPopup(Object.keys(feature.properties).map(function(k) {
                        return k + ": " + feature.properties[k];
                    }).join("<br />"), {
                        maxHeight: 200
                    });
                }
            }
        });
            shpfile.once("data:loaded", function() {
            console.log("finished loaded shapefile");
        });
```

You can click on a feature to read the attribute table.

3.3. Minimap

Minimap plugin allows you to display a dynamic situation map in a corner of your map. A situation map is intended to give an overview of current extent.

Download the plugin : https://github.com/Norkart/Leaflet-MiniMap/archive/master.zip

Extract the .zip in a minimap and try by yourself to add a minimap.

Be carfeull, you cannot use the same base layers in your main map and in your minimap. You have to define a new base layers especially for the minimap.

3.4. Geocoding

Geocoding typically refers to the transformation process of addresses and places to coordinates, and is sometimes called forward geocoding whereas Reverse geocoding uses geographic coordinates to find a description of the location, most typically a postal address or place name.

https://github.com/k4r573n/leaflet-control-osm-geocoder

We are going to add a geocoding function :

```
<!--calling script Geocoder -->
<script src="http://k4r573n.github.io/leaflet-control-osmgeocoder/Control.OSMGeocoder.js"></script>
```

Calling the css :

```
        <!-- stylesheet geocoder-->
<link rel="stylesheet" href="http://k4r573n.github.io/leaflet-control-osm-geocoder/Control.OSMGeocoder.css" />
```

Add the control to the map :

```
//Ajout Geocoder
var osmGeocoder = new L.Control.OSMGeocoder();
map.addControl(osmGeocoder);>
```

You will see a text field where you can type a location in order to zoom on it.

3.5. Draw

This plugin is usefull to draw some geometric elements (points, lines, polygons) in your map.

https://github.com/Leaflet/Leaflet.draw

In a complex way, it is possible to upload, save and store these draws in a database.

3.6. Base Layers

You must be now in capacity to add many base layers. You can use these plugins to do that and try to add mapbox, bing, google maps base layers. For Bing and Google, you need an API Key by signing-in directly on the official websites.

https://github.com/shramov/leaflet-plugins

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

3.7. Marker Cluster

https://github.com/Leaflet/Leaflet.markercluster

3.8. Add your own data

http://makinacorpus.github.io/Leaflet.FileLayer/

3.9. Graticule

https://github.com/turban/Leaflet.Graticule

3.10. Display OSM Data

OpenStreetMap (OSM) is a collaborative project to create a free editable map of the world.

The next plugin allows you to display objects you want by using overpass api.

https://overpass-turbo.eu/

A list of features is avalaible here :

http://wiki.openstreetmap.org/wiki/Map_Features

With the plugin, we are going to display Football playgrounds here in Zanzibar.

Download the plugin, see the demos pages and try to implement.

https://github.com/kartenkarsten/leaflet-layer-overpass/

Another plugin is usefull to deal with OpenStreetMap data :

https://github.com/yohanboniface/Leaflet.RevealOSM

3.11. Location list

This plugin allows you to zoom to a specific area. It displays a control which scrolls the map through the list of locations and zooms.

https://github.com/mithron/leaflet.locationlist

Download the plugin and try to implement the showList function in order to show a select box for quick jumps to the locations.

3.12. CartoDB Layer

CartoDB is a cloud-based solution for mapping.

First of all, you have to create an account http://cartodb.com/

Registration is free but the account is limited to 5 tables, 5 Mb and 10K map views.

A 14-day trial access is possible.

CartoDB provides some tools to create and store layers on the cloud.

By the way, it is possible to display with Leaflet your CartoDB layers.

3.13. D3.js Integration

3.14. WFS-T

https://github.com/Leaflet/Leaflet.Editable

4. Deployment

4.1. On your local Web Server

Install a web server like Apache HTTP Server, nginx, XAMPP or another one.

Copy your files in the htdocs directory in order to access to your application with http protocol.

http://localhost/webmap

4.2. On the web

There are many free providers to host online your application.

Sign in and configure an FTP client to upload your application on the web :

http://www.5gbfree.com

or choose another one here :

20 Web Hostings Sites