Προγραμματισμός και Συστήματα στον Παγκόσμιο Ιστό

JavaScript - LeafletJS

Δρ. Δημήτριος Κουτσομητρόπουλος

Πρόσβαση σε χάρτες

JavaScript Map APIs

- Βιβλιοθήκες που παρέχουν αντικείμενα και συναρτήσεις για πρόσβαση, εμφάνιση και παραμετροποίηση χαρτογραφικής πληροφορίας
- Γνωστές βιβλιοθήκες:
  - Google Maps API 
  - OpenLayers 
  - LeafletJS 

Map APIs

Δεν περιέχουν την χαρτογραφική πληροφορία!
- Δεν έχουν χάρτες
- Μπορούν να επικοινωνήσουν με παρόχους χαρτών για τη φόρτωσή τους

Φορτώνουν τους χάρτες με τη μορφή πλακιδίων (tiles)
- Ένας χάρτης αποτελείται από έναν αριθμό tiles

Πάροχοι χαρτών:
- Google Maps
- OpenStreetMap (OSM)
  - Συνεργατική ανάπτυξη χαρτών με ελεύθερη άδεια
- MapBox
- Apple maps
- …

LeafletJS

www.leafletjs.com

Πλεονεκτήματα:
- Ανοιχτού κώδικα
  - https://github.com/Leaflet/Leaflet
- Μικρού όγκου (~38KB)
- Κατάλληλο για κινητές συσκευές
- Πολυάριθμα plugins
  - Π.χ. για φόρτωση KML αρχείων, heatmaps, αναζήτηση, POIs…
- Δυνατότητα προσθήκης επιπέδων
  - Σημεία ενδιαφέροντος, σχήματα, πολύγωνα
- Ανεξάρτητο παρόχου χαρτών

Χρήση

Εισαγωγή του Leaflet CSS στο έγγραφο:

```
<link rel="stylesheet"
href="https://unpkg.com/leaflet@1.3.4/dist/leaflet.css"
/>
```

- Περιέχει τις πληροφορίες εμφάνισης του πλαισίου του χάρτη, των κουμπιών ζουμ κλπ

Εισαγωγή της JS βιβλιοθήκης μετά το CSS:

```
<script
src="https://unpkg.com/leaflet@1.3.4/dist/leaflet.js">
</script>
```

- Περιέχει το αντικείμενο **L**, που δίνει πρόσβαση στις λειτουργίες της βιβλιοθήκης, π.χ.
  `L.map(), L.setView()…`

Χρήση

Δημιουργούμε ένα στοιχείο στη σελίδα που θα φιλοξενήσει το χάρτη, με συγκεκριμένο id:

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

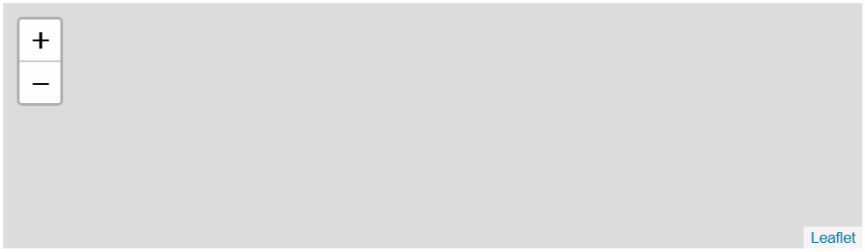
Ορίζουμε ένα μέγεθος στο CSS:

```
#mapid { height: 180px; }
```

Δημιουργία χάρτη:

```
let mymap = L.map('mapid');
```

Αποτέλεσμα εκτέλεσης



Δεν εμφανίζεται χάρτης!

Θα πρέπει να φορτωθούν τα **tiles**

Προσθήκη tiles

Θα πρέπει να δημιουργηθεί ένα TileLayer και να προστεθεί στον χάρτη:

```
let tiles =
L.tileLayer('http://{s}.somedomain.com/{foo}/{z}/{x}/{y}.png?
{foo}', {foo: 'bar', attribution: '&copy;…'})
```

Options obj

- URL: με { } καθορίζεται η δομή του URL του παρόχου
  - {z} το επίπεδο zoom
  - {x}, {y}: συντεταγμένες
  - {s}: subdomain για parallel requests
  - foo: Παράμετροι GET που θέλουμε να περάσουμε στο URL (π.χ. attribution)
- Παράδειγμα URL για OpenStreetMap:

```
https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png
```

- Προσθήκη του layer στο χάρτη:

```
map.addLayer(tiles); ή tiles.addTo(mymap);
```

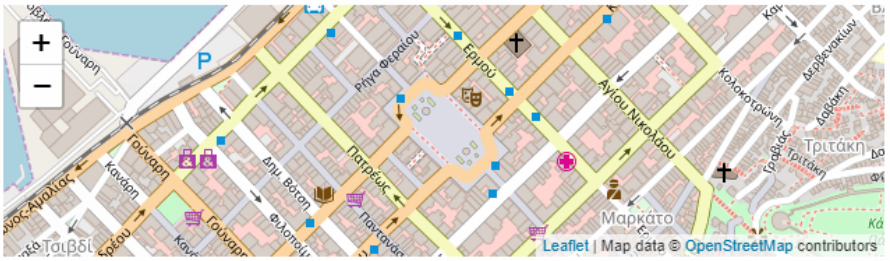
Εμφάνιση χάρτη

Δεν αρκεί το addLayer για να εμφανιστεί ο χάρτης!

Χρειάζεται να οριστεί *σε ποιο σημείο θέλουμε να εμφανιστεί το κέντρο του χάρτη (συντεταγμένες και zoom)*:

```
mymap.setView([38.2462420, 21.7350847], 16);
```

codepen



Προσθήκη σημείων στο χάρτη

Δημιουργία σημείου:

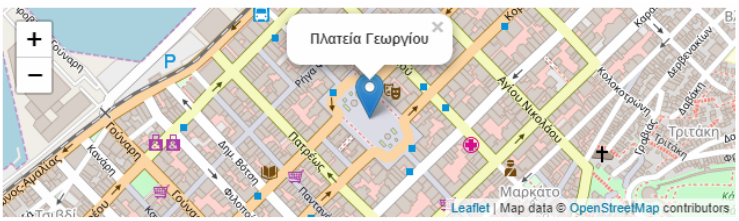
```
let marker = L.marker ([38.246242, 21.7350847]);
```

Προσθήκη στο χάρτη:

```
marker.addTo(mymap);
```

Προσθήκη popup:

```
marker.bindPopup("<b>Πλατεία Γεωργίου</b>");
```



codepen

Σχεδίαση πολυγώνων

```
let points = [[38.246598,
21.736236],[38.24723,
21.737019],[38.246733,
21.737652],[38.246101, 21.736836]];

let polygon = L.polygon(points,
{color:"red", fillColor:
"red"}).addTo(mymap);

let center =
polygon.getBounds().getCenter();

mymap.setView(center, 18);
```

Λίστα με τις συντεταγμένες των κορυφών του πολυγώνου

Δημιουργία του πολυγώνου και χαρακτηριστικά του

Εύρεση κεντροειδούς πολυγώνου

Εστίαση και ζουμ

codepen

Events

```
let marker = L.marker([38.246242,
21.7350847], { draggable: "true" });

marker.addTo(mymap);

marker.bindPopup("<b>Πλατεία
Γεωργίου</b>").openPopup();

marker.on("click", markerClick);

function markerClick(event) {
      this.getPopup()
      .setLatLng(event.latlng)
      .setContent("Συντεταγμένες
σημείου: " + event.latlng.toString());

}
```

Ο marker είναι ***draggable***

Προσθήκη eventHandler. Μέθοδος **on()**: όπως η *addEventListener*

Το **this** αναφέρεται στο ***αντικείμενο που προκάλεσε το event*** (marker). Το event αντικείμενο περιέχει χρήσιμες πληροφορίες για το event

Method chaining: οι μέθοδοι επιστρέφουν το ίδιο το αντικείμενο

codepen

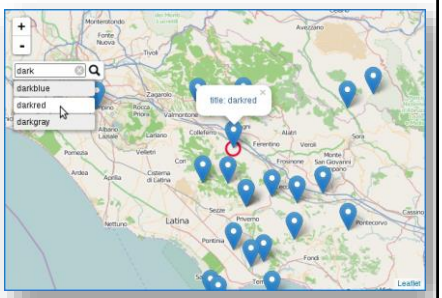
Αναζήτηση (L.Control.Search)

Αναζήτηση markers

- Βάσει ονόματος, ιδιότητας, κατηγορίας…
- Εμφάνιση πεδίου ελέγχου (Control) πάνω στο χάρτη

Leaflet.Control.Search

- Plugin του LeafletJS για πεδίο ελέγχου αναζήτησης
- Ανοιχτού κώδικα
  - https://github.com/stefanocudini/leaflet-search
- Απλό στη χρήση
- Δυνατότητα autocomplete



Χρήση

Εισαγωγή της JS βιβλιοθήκης

- Εγκατάσταση στο server με npm
- Εναλλακτικά από κάποιο cdn:

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/leaflet-
search/3.0.2/leaflet-search.min.js"> </script>
```

- Περιλαμβάνει το αντικείμενο L.Control.Search που αντιστοιχεί στο πεδίο αναζήτησης
- **L.Control** κλάση βάσης του Leaflet για όλα τα controls

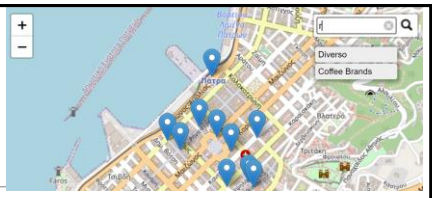
Εισαγωγή του CSS:

```
<script src="https://cdn.jsdelivr.net/npm/leaflet-
search@3.0.2/dist/leaflet-search.min.css"> </script>
```

- Καθορίζει πώς εμφανίζεται το control
- Περιλαμβάνει τυχόν icons (π.χ. Search icon)

Παράδειγμα

codepen




```
let data = [{loc: [21.7352181, 38.2466877],
title: "Zizu"},…];
let markersLayer = new L.LayerGroup();
mymap.addLayer(markersLayer);
let controlSearch = new L.Control.Search({
   position: "topright",layer:markersLayer,
   initial: false, zoom: 15,
   marker: false});
mymap.addControl(controlSearch);
for (i in data) {
  let title = data[i].title;
  let loc = data[i].loc;
  let marker = L.marker(L.latLng(loc),
    {title: title});
  marker.bindPopup("title: " + title);
  marker.addTo(markersLayer);
}
```

LayerGroup: Ομαδοποιεί Layers. Ένας marker *είναι* Layer.

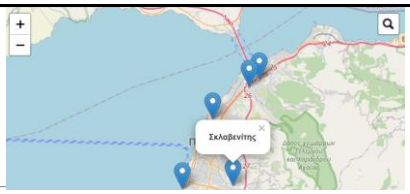
Το configuration του Control.Search. **layer**: πού θα γίνει η αναζήτηση. **propertyName:** σε *ποιο πεδίο* θα γίνει η αναζήτηση (default: 'title')

Προσθήκη του control στο χάρτη

Δημιουργία markers για τα σημεία ενδιαφέροντος. Options obj: *icon, **title**…* Μπορούν να προστεθούν και custom properties.

Προσθήκη των σημείων στο LayerGroup.

Φόρτωση GeoJSON




```
…
let data = {"type": "FeatureCollection",
  "features": […] };

let featuresLayer = new L.GeoJSON(data, {
  onEachFeature: function (feature, marker)
{
    marker.bindPopup("<h4>" +
    feature.properties.name + "</h4>");
  }
});
featuresLayer.addTo(mymap);
…
```

The *onEachFeature* option is a function that gets called on each feature before adding it to a GeoJSON layer

```
{
 "type": "FeatureCollection",
 "features": [
  {
   "type": "Feature",
   "properties": {
    "type": "Point",
    "coordinates": [
     21.712654,
     38.2080319
    ]
   },
   "id": "node/354449389"
  },…]
}
```

Codepen

Λίστα παραδειγμάτων