15

Créer un plugin Leaflet

15.1. Travailler avec plusieurs points

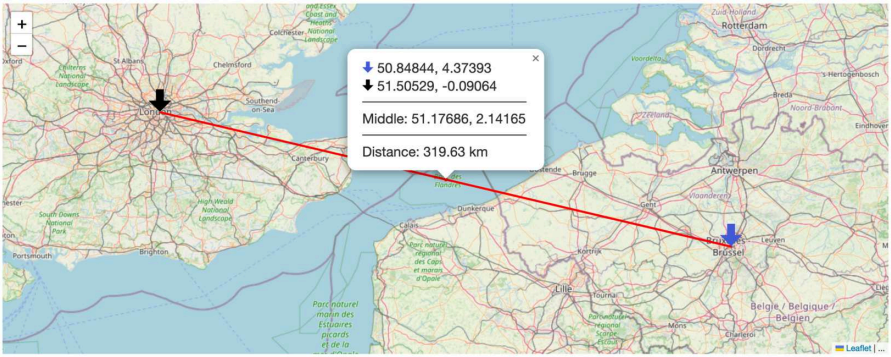
Leaflet possède une fonction `distance` qui permet de calculer la... distance (!) entre deux points, en tenant compte de la courbure de la planète (qui, rappelons-le, n'est pas plate). Ultra simple d'emploi, elle tient en une ligne, supposant que `p1` et `p2` soient, bien sûr, des points définis.

```
p1.distanceTo(p2)
```

Pour la mettre en œuvre, nous allons rechercher une certaine difficulté et profiter de l'occasion pour aborder la création d'une extension Leaflet, susceptible d'être ajoutée à n'importe quelle carte. Nous en verrons le principe et les bases appliquées à notre besoin (afficher la distance entre deux points), sans creuser plus que nécessaire.

Le plugin complet (et légèrement amélioré) est disponible sous le nom de `Leaflet.MesureDistance` sur mon Github personnel.

Figure 15.1 : *La version du plugin en ligne, nous allons en faire un plus simple...*



15.2. Pourquoi passer par un plugin ?

D'une manière générale, le fait de passer par un plugin offre plusieurs avantages. Citons par exemple :

- *Réutilisabilité* : il est plus facile de le réutiliser dans un autre contexte, sur un autre projet. Voire de le partager avec la communauté.
- *Maintenabilité* : le code étant isolé, sa maintenance est plus aisée et ne perturbe pas le reste de l'application.
- *Modularité* : chaque fonctionnalité clé sera séparée dans un plugin, sans remettre en cause le cœur de notre application.
- *Testabilité* : le plugin sera testable (et testé) de façon indépendante, là encore sans perturber le reste de l'application (il sera bien sûr également testé avec le reste).

Il y aura également quelques inconvénients :

- *Complexité initiale* : il est plus complexe, plus abstrait de raisonner à l'échelle d'un plugin.
- *Configurabilité* : il faut raisonner en termes d'options que l'on va pouvoir appliquer ou non, ce qui complexifie potentiellement le code.
- *Documentation* : surtout si vous le diffusez, créer une documentation (en anglais si possible) sera toujours une nécessité. Pas le plus complexe, mais peut prendre un peu de temps.

Bien sûr, encore une fois, le nôtre sera particulièrement simple.

15.3. Qui veut voyager loin, prépare sa monture…

Le premier point est bien entendu de définir ce que va faire notre plugin, et voir quels sont les paramètres qui lui seront nécessaires pour fonctionner.

Puisque notre outil va servir à mesurer une distance entre deux points, il va falloir créer deux marqueurs. Notre plugin aura donc deux paramètres nécessaires : les coordonnées des deux points.

Par précaution, nous définirons des valeurs par défaut, qui s'appliqueront en absence de paramètres. Une version "luxe" (disons plutôt "de production") ajoutera

des contrôles de validité de ces paramètres (s'assurer que ce soient bien des coordonnées), aspect que nous mettons de côté ici.

À réception des paramètres, le plugin va alors créer deux marqueurs, correspondant respectivement au point A et au point B, qui seront `draggable`. Premier problème, seuls les marqueurs sont déplaçables, pas les cercles ou rectangles. Et Leaflet ne propose qu'un seul marqueur (bleu) par défaut, rendant impossible l'identification de l'un ou de l'autre... Nous pourrions utiliser des marqueurs personnalisés, nécessitant des images externes (qui peuvent devenir des options) ou des images en Data URI (encodées en base64 et stockées directement dans le code), options que j'ai retenues dans la version distribuable du plugin.

Dans notre version, pour simplifier, nous afficherons le popup avec les infos sur un des deux points, ce qui permettra de le distinguer (important surtout si on affiche aussi les coordonnées de chaque point, aucun intérêt en soi si on n'affiche que la distance).

Une fois les marqueurs placés, notre plugin va donc tracer une ligne entre les deux, mesurer la distance et actualiser le popup. Le tout étant actualisé en cas de déplacement de l'un ou l'autre des marqueurs.

15.4. L'extension MeasureDistance

Dans l'idée d'isolation du code évoquée plus haut, nous allons créer un fichier à part pour notre plugin. Nommons le `measure.js`.

Un plugin est une extension des classes de base de Leaflet, c'est-à-dire `L.Layer`, `L.Handler` ou `L.Control`, et doit retourner une instance de la classe, via une structure de ce genre. Notez l'utilisation des majuscules et minuscules pour distinguer les instances de la classe.

```
L.Control.MeasureDistance = L.Control.extend({
  // @todo
});
L.control.measureDistance = function (options) {
    return new L.Control.MeasureDistance(options);
};
```

La première étape sera de définir les options et leurs valeurs par défaut. La méthode `onAdd` s'exécutera automatiquement à l'ajout sur la carte, le pendant du `addTo(map)`