7

Marqueurs, popups

7.1. Ajouter un marqueur et une fenêtre d'information

Notre carte est magnifique, mais encore bien vide. Elle est certes centrée sur un point, mais nous ne voyons pas ce point. Nous allons donc ajouter un marqueur. Pour y aller progressivement, dupliquons notre fichier `101.html` et son `101.js` en 102 (attention à modifier le `src` du JavaScript), et ajoutons ce marqueur.

Pour cela, à la suite de notre construction et après avoir ajouté le layer, il suffit de créer une instance de la classe `marker`, en précisant sa position (tableau ou objet), sans oublier de l'ajouter à la carte !

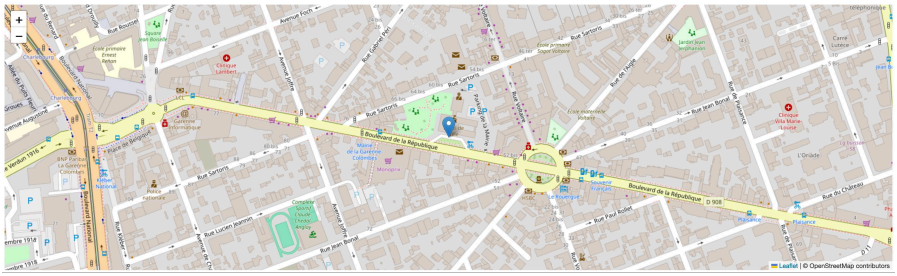
```
let center = [48.9068, 2.2464];

// création de la carte, code que nous ne reprendrons pas
// de façon systématique
let map = L.map("map");
map.setView(center, 17);
let layer = L.tileLayer(
  "https://tile.osm.org/{z}/{x}/{y}.png",
  { attribution: `&copy; OpenStreetMap contributors` }
);
layer.addTo(map);

// ajout de marqueur
const marker = L.marker(center);
marker.addTo(map);
```

Créer une variable `layer` ne sert pas réellement, à moins de vouloir agir sur ce layer (le cacher pour en montrer un autre par exemple). Je le mets assez souvent, plus par habitude. Pour `marker` non plus, si vous utilisez le chainage, la variable n'est pas indispensable (mais rend le code souvent plus clair).

Figure 7.1 : *Mieux, surtout en modifiant le niveau de zoom*



Leaflet accepte le chainage des éléments dans de nombreux cas. Il est donc possible de faire l'ajout de l'élément directement (aussi bien le marqueur que les tuiles ou le `setView`).

```
let center = [48.9068, 2.2464];
let map = L.map("map").setView(center, 17);
L.tileLayer("https://tile.osm.org/{z}/{x}/{y}.png",
  { attribution: `&copy; OpenStreetMap contributors` }
).addTo(map);
const marker = L.marker(center).addTo(map);
```

Notez qu'ici, pas de variable `layer`.

Notre marqueur étant souvent un élément d'information, nous pouvons lui ajouter une fenêtre interactive, qui s'ouvre si on clique dessus. Dans le vocabulaire Leaflet, nous parlerons de *popup*.

```
marker.bindPopup("La Mairie");
```

Nous pouvons également faire en sorte qu'il s'ouvre automatiquement au chargement de la carte.

```
marker.bindPopup("La Mairie").openPopup();
```

Notez que nous pouvons également créer des popups indépendants, qui ne sont pas liés à un marqueur. Il suffit alors d'instancier un objet de type `L.popup()`, de lui donner des coordonnées et un contenu, et de l'ouvrir.

```
let popup = L.popup()
  .setLatLng([48.9064, 2.2483])
  .setContent("Place de la liberté")
  .openOn(map);
```

Vous constatez également que le marqueur indépendant, une fois fermé, ne peut plus s'ouvrir manuellement, car nous n'avons pas d'endroit où cliquer... Nous pourrions par contre le rouvrir via du code, en rappelant `popup.openOn(map);`.

À l'ouverture de notre deuxième popup, le premier s'est refermé. Un seul popup à la fois peut être ouvert, à moins de préciser que l'on ne veut pas que ce soit le cas. Popups comme marqueurs disposent de nombreuses options. Ici notre marqueur de piscine sera non fermable (`autoClose` et `closeOnClick` à `false`), et nous retirons le bouton de fermeture.

```
let piscine = L.popup({
  autoClose: false,
  closeOnClick: false,
  closeButton: false,
}).setLatLng([48.9053, 2.2437])
  .setContent("Piscine")
  .openOn(map);
```

Si nous voulions le faire au niveau du marqueur de la mairie, ce serait le `bindPopup` qui recevrait les options.

```
const marker = L.marker(center).addTo(map);
marker.bindPopup("La Mairie", {
    autoClose: false,
    closeOnClick: false,
    closeButton: false,
}).openPopup();
```

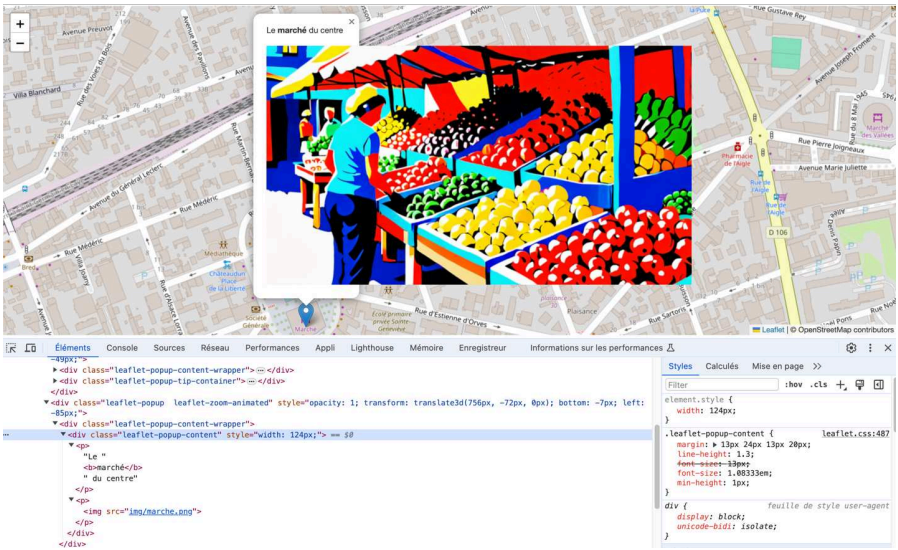
Attention *› Prenez garde quand même à ne pas multiplier les popups persistants, cela encombre vite une carte.*

Le contenu d'un popup peut être du texte, mais également du texte formaté en HTML, voire une image (dans notre cas, générée par une IA, Stable Diffusion).

```
const marche = L.marker([48.909, 2.2468]).addTo(map);
marche.bindPopup(`
  <p>Le <b>marché</b> du centre</p>
  <p><img src="img/marche.png" /></p>
  `).openPopup();
```

Mais attention aux dimensions ! Quand Leaflet génère le popup, il en résulte une batterie d'éléments HTML, avec de nombreuses classes. À nous de les exploiter dans notre CSS.

Figure 7.2 : *On remarque que l'image déborde*



L'inspecteur du navigateur va nous renseigner sur la classe à utiliser pour pouvoir cibler les images, par exemple en ajoutant cette ligne dans notre CSS :

```
.leaflet-popup-content img { max-width: 100%; }
```

Un autre moyen étant soit de mettre une classe sur la balise `img`, soit via le paramètre `className` de notre popup. Citons au passage d'autres options intéressantes du popup permettant de le dimensionner : `maxWidth`, `minWidth`, `maxHeight`.

```
const marche = L.marker([48.909, 2.2468]).addTo(map);
marche.bindPopup(`
  <p>Le <b>marché</b> du centre</p>
  <p><img src="img/marche.png" class="img-popup" /></p>
  `,{
    maxWidth: 300, minWidth: 300, maxHeight: 150,
    className: "scroll-popup",
  }).openPopup();
```