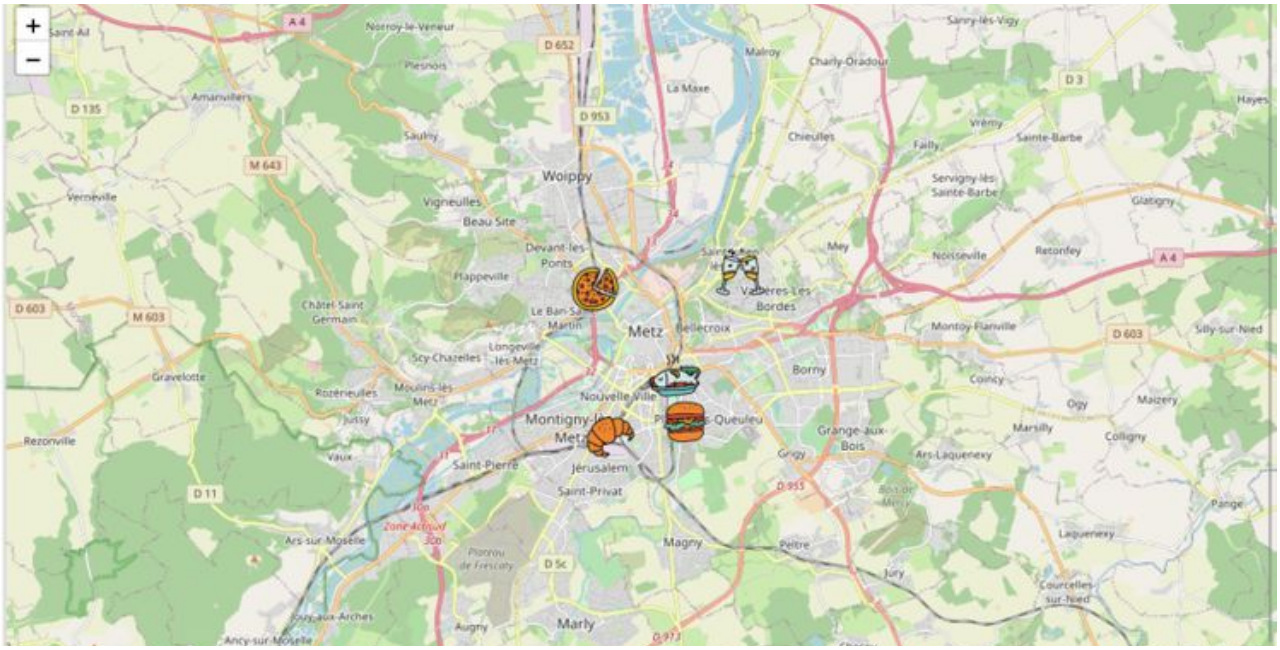
29 DÉC. 2020 • 5 MIN READ • VUE.JS

Créez vos cartes interactives avec Vue.js et Leaflet

Nous allons, au cours des prochains articles, apprendre ensemble à créer un site web permettant l'affichage d'une carte de nos restaurants favoris, le tout grâce au framework Vue.js et à Leaflet.



Bonjour et bienvenue à tous sur ce tutoriel. Nous allons, au cours des prochains articles, apprendre ensemble à créer un site web permettant l'affichage d'une carte de nos restaurants favoris, le tout grâce au framework Vue.js et à Leaflet.

Introduction et Setup

Vue.js

Comme dit précédemment, nous allons utiliser Vue.js. Si vous ne le connaissez pas, il s'agit d'un framework Javascript open-source très populaire, qui tient l
Angular et React. Je vous invite à aller vous renseigner dessus à cette a
https://vuejs.org/

Leaflet

Leaflet est une librairie Javascript permettant de créer des cartes interactives et customisables. Librairie très populaire, facile à utiliser, elle sera au centre de ce tutoriel. Il existe une version spécifique pour Vue.js: **https://vue2-leaflet.netlify.app/**

Générer le projet

```
npm install -g @cli/vue
vue create npmduprojet
```

Pour lancer le projet, on utilisera la commande:

```
npm run serve
```

Création de notre premier composant

Rendez vous tout d'abord dans le projet généré, et allez dans le dossier src/components et créez un nouveau fichier Vue. Appelons le map.vue

On le bind alors dans src/App.vue (cf exemple), avant de nous attaquer à remplir ce fichier map.vue.

```
<template>
 <div id="app">
   <Map/>
 </div>
</template>

<script>
import Map from './components/map.vue'

export default {
 name: 'App',
 components: {
   Map
```

```
 }
}
</script>

<style>
</style>
```

A partir de ceci, on va alors se demander ce que l'on veut voir s'afficher: Une carte bien sûr, puis des icônes s'affichant dessus à des coordonnées précises et chacune ayant son design propre ou ses propres informations lorsqu'on clique dessus... Commençons donc par afficher la carte.

Pour cela, nous allons donc utiliser Leaflet, qui va nous servir donc à effectuer le rendu d'une carte dans notre application. Les choses étant bien faites, il existe une version de Leaflet spécialement pour Vue.js: Vue2Leaflet. Nous allons donc installer via npm les librairies nécessaires. A vos lignes de commandes:

```
npm install --save leaflet
npm install --save vue2-leaflet
```

Il est très important d'installer Leaflet car il ne sera pas installé automatiquement lors de l'installation de vue2-leaflet. On s'exposerait alors à des troubles de fonctionnement assez évitables.

Voici les 3 éléments que nous allons utiliser et leur fonctionnement:

l-map: Le composant de base, contient tous les autres éléments et permet de gérer les principaux paramètres.

l-tile-layer: Contient l'url de la map, ainsi que des options telles que le niveau de zoom ou la position par défaut.

l-marker: Composant contenant les marqueurs qui apparaitront sur la map. Dans notre cas, le marker contiendra les icones des restaurants.

```
<template>
 <l-map

   :center="center"
```

```
      :center="center"
      :zoom="zoom"
      class="map"
      ref="map"
      @update:zoom="zoomUpdated"
      @update:center="centerUpdated"
    >
      <l-tile-layer
        :url="url"
      >
      </l-tile-layer>
      </l-map>
</template>

<script>
import { LMap, LTileLayer } from 'vue2-leaflet';
import 'leaflet/dist/leaflet.css';

export default {
 components: {
   LMap,
   LTileLayer
 },
 data () {
   return {
     url: 'https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png',
     center: [ 49.1193089, 6.1757156 ],
     zoom: 12,
   }
 },
 methods: {
   zoomUpdated (zoom) {
     this.zoom = zoom;
     console.log(this.markers)
   },
   centerUpdated (center) {
     this.center = center;
   }
 }
}
</script>

<style>

  map {
```

```
.map {
  position: absolute;
  width: 100%;
  height: 100%;
  overflow :hidden
 }
</style>
```

Dans cet exemple, on voit ainsi la configuration “de base” permettant l’affichage de la carte dans notre navigateur, sans ajouts de marqueurs.

Nous allons maintenant ajouter ce qui rendra notre carte unique: les l-marker. Comme expliqué plus haut, Ils seront placés sur la carte à des points précis afin, comme leur nom indique de marquer un emplacement. On les place juste après le l-tile-layer. Voici ci dessous le code associé au design:

```
<l-marker
 :key="marker.id"
 :lat-lng="marker.coordinates"
>
</l-marker>
```

On rajoute également dans data():

```
markers: [
   {id: 1, coordinates: [ 49.114910, 6.178810 ]},
   {id: 2, coordinates: [ 49.133290, 6.154370 ]},
   {id: 3, coordinates: [ 49.102160, 6.158850 ]},
   {id: 4, coordinates: [ 49.136010, 6.199630 ]},
   {id: 5, coordinates: [ 49.105563, 6.182234 ]},
]
```

Comme on le voit, tout fonctionne et les marqueurs sont présents.

Satisfaits? Non? Il est vrai que le design de base laisse clairement à désirer. Heureusement, il est possible de le changer et de customiser les icônes de ces marqueurs afin qu’ils soient plus adaptés à ce qu’on veut leur faire montrer. Il est pour cela possible de créer une icône personnalisée. Afin de rendre notre code lisible et de pouvoir réutiliser ce qui sera l'icône de nos restaurants, nous allons créer un

et de pouvoir réutiliser ce qui sera l'icône de nos restaurants, nous allons créer un composant à part qui contiendra la future icône:

```
<template>
 <l-marker
   :key="marker.id"
   :lat-lng="marker.coordinates"
 >
   <l-icon ref="icon">
     <img class="restaurant-icon" :src="marker.imageUrl"/>
   </l-icon>
 </l-marker>
</template>

<script>
import { LIcon, LMarker } from 'vue2-leaflet'
export default {
 components: { LIcon, LMarker },
 props: {
   marker: {
     type: Object,
     required: true
   }
 }
}

</script>

<style>
 .restaurant-icon {
   height: 50px;
   width: auto;
 }
</style>
```

Ne pas oublier de l'appeler à la place du l-marker dans map.vue

On note la présence de l-icon: Ce composant permet de construire un composant Vue qui sera utilisé pour créer l'icône: Permet interactivité / code propre, voir de construire des icônes complexes (Plusieurs images qui se chevauchent, dynamisme). On peut également faire l'inverse et mettre le composant dans le l-icon afin de pouvoir réutiliser l'image icône dans un autre contexte, tout dépend des choix de

pouvoir réutiliser l'image icône dans un autre contexte, tout dépend des choix de développement. Dans le l-icon est précisée l'image qui lui sera associée, qu'on a définie dans les markers mis à jour:

```
markers: [
        {id: 1, imageUrl: 'https://img.icons8.com/doodle/48/000000/fish
        {id: 2, imageUrl: 'https://img.icons8.com/doodle/48/000000/pizz
        {id: 3, imageUrl: 'https://img.icons8.com/doodle/48/000000/croi
        {id: 4, imageUrl: 'https://img.icons8.com/doodle/48/000000/the-
        {id: 5, imageUrl: 'https://img.icons8.com/doodle/48/000000/hamb
      ]
```

Et voici donc le design de notre application:

Vous pouvez retrouver l'intégralité du code de ce projet ici:

Jorek57/VueLeaflet-Tutorial

Contribute to Jorek57/VueLeaflet-Tutorial development by creating an account on GitHub.

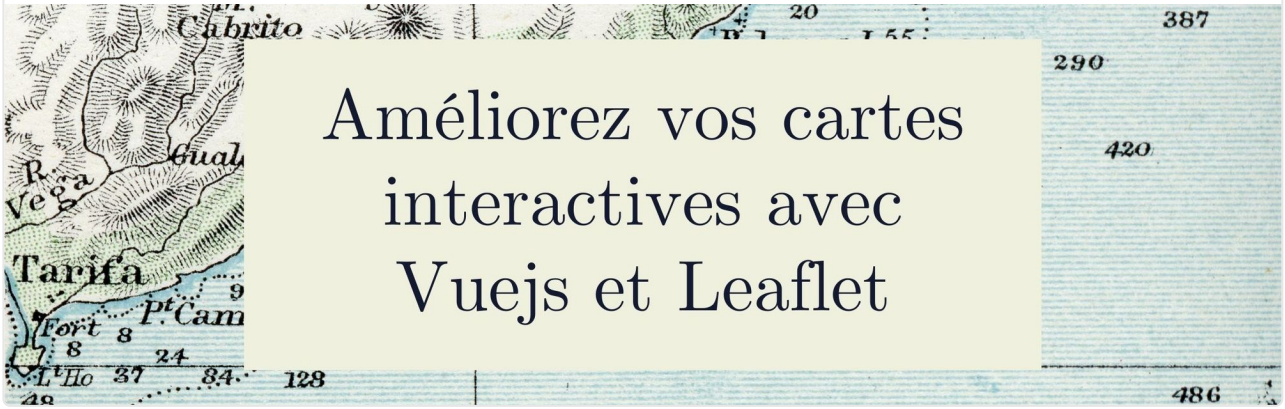
GitHub • Jorek57

Dans la seconde partie, nous apprenons à créer des clusters d'icône, et à améliorer l'UX de notre carte!

Améliorez vos cartes interactives avec Vuejs et Leaflet

Dans l'article précédent, nous avons appris à faire apparaitre des marqueurs sur une carte. Le…

NK Nicolas KEMPF • Kevin Cluzel



Vous aimez mes articles? Inscrivez-vous pour être alerté des nouveautés.

S'INSCRIRE

Déja inscrit? Connectez-vous

Publié par:

SHARE TWEET

Vous aimerez aussi...

FÉVR. 28 Améliorez vos cartes interactives avec Vuejs et Leaflet 6 min read