

Tutoriel Vue.js

Introduction

Vue est un framework javascript permettant de développer des interfaces utilisateurs. Comme ses principaux concurrents React et Angular, Vue se veut être un framework visant à simplifier et organiser le développement d'applications web en implémentant un bon nombre de fonctionnalités telles que les modèles, les composants, les transitions, le routage, etc...

La particularité de Vue.js est qu'il a été créé pour permettre des développements rapides tout en possédant une simplicité d'utilisation et de compréhension de la logique. Ainsi migrer depuis un projet sans framework est beaucoup plus facile. Un composant au sens Vue.js est représenté par un fichier qui a comme extension « .vue », dans ce fichier on peut retrouver le modèle du composant (son template html, son style css mais aussi sa logique métier grâce à un script javascript).

Les vues

Tout comme ses concurrents React Native et Angular, on peut réutiliser un même composant en définissant lors de sa création les données qu'il contiendra. Un fichier vue est découpée en 3 parties correspondant au code html, js et css.

```
<template>
  <div>
    <div v-for="(run, index) in runs" :key="index" v-on:click="clickRun(index)">
      <Run :title=run.title :description=run.description :date=run.date :img=run.img />
    </div>
  </div>
</template>
```

```
<script>
import RunsJSON from "../resources/runs.json";
export default {
  name: "Runs",
  components: {
    Run
  },
  data: function() {
    return {
      runs: RunsJSON.runs,
    };
  },
  methods: {
    clickRun: function(index) {
      this.clickedItem = index;
      this.$emit("clicked", index);
    }
  }
};
</script>
```

```
<style scoped>
.navbar-brand {
  background: #006b00;
  width: 100%;
  color: white;
  padding-right: 0px;
  margin-right: 0px;
  cursor: pointer;
}
</style>
```

Figure 1 : Exemple type d'un fichier Vue.js

Le framework vue dispose de plusieurs fonctions et directives permettant de simplifier le développement des composants. Il existe par exemple la directive `v-if='isValid()'` qui est une structure conditionnelle d'affichage, l'élément sera affiché seulement si la condition est vérifiée. J'ai pu utiliser également `v-for='(run, index) in runs'` qui permet le rendu d'une liste d'éléments avec la notion d'élément courant et son index. Ainsi comme le montre la figure 1, il est très simple de définir le composant fils qui sera réutilisé dans la boucle du v-for.

Les évènements

L'écriture d'évènement avec Vue.js est rendue très accessible et simple d'implémentation. En effet il est possible de définir sur n'importe quel élément une directive de type `v-on :click=addMarker()'`

```
<div v-for="(run, index) in runs" :key="index" v-on:click="clickRun(index)">
  <Run :title=run.title :description=run.description :date=run.date :img=run.img />
</div>
```

```
methods: {
  clickRun: function(index) {
    this.clickedItem = index;
  }
}
```

Communication entre les composants

Ma principale difficulté dans la prise en main de ce framework a été de bien gérer la communication entre les composants. Par exemple dans l'application, sur la vue de mes tracés, dès lors que l'on clique sur un tracer en particulier, la carte est mise à jour pour afficher le tracer. Ainsi je dois faire remonter l'information au parent comme quoi un des tracés a été cliqué et ensuite le parent doit informer le composant GoogleMap.vue pour que la carte se mette à jour. La documentation officielle de Vue.js préconise d'utiliser les \$emits et les évènements.

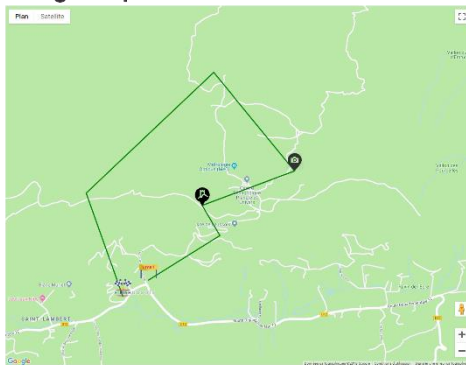
Navbar.vue

PATH PARTOUT

Runs.vue

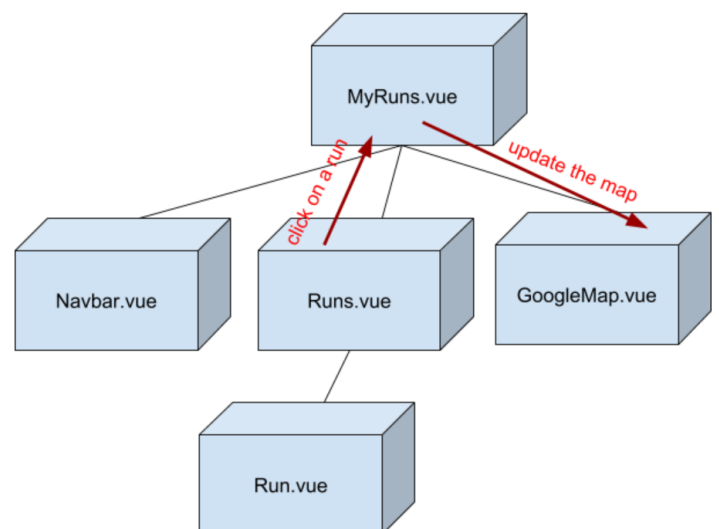


GoogleMap.vue



Run.vue

Run.vue



J'ai donc procédé comme suit :

Runs.vue

```
<div v-for="(run, index) in runs" :key="index" v-on:click="clickRun(index)">
  <Run :title=run.title :description=run.description :date=run.date />
</div>
```

```
clickRun: function(index) {
  this.clickedItem = index;
  this.$emit("clicked", index);
},
```

MyRuns.vue

```
<div class="col-md-4">
  <Runs @clicked="onClickChild"/>
</div>
<div class="col-md-8">
  <div>
    <GoogleMap ref="runIndex"/>
  </div>
</div>
```

```
onClickChild(value) {
  this.$refs.runIndex.setMarkers(value);
},
```

GoogleMap.vue

```
setMarkers(index) {
  this.path = this.runs[index].path;
  this.markers = this.runs[index].pois;
},
```

Référencement des avantages et des inconvénients

Avantages

- Architecture modulaire
- Facilité de migration d'un projet sans framework vers Vue.js
- Framework très flexible
- Contrairement à React, Vue.js est lié au DOM et utilise des attributs HTML spéciaux pour rendre le DOM réactif
- Poids du framework comparé à React et Angular

Inconvénients

- Framework récent et peu documenté
- Fichiers « .vue » qui peuvent être rapidement énormes puisqu'ils embarquent le code HTML, CSS et JS
- Manque de ressources
- Communication entre les composants assez lourde (voir la partie correspondante ci-dessus)