

VueJS

Analyse de technologie - PRW3

Par Eric Bousbaa



Introduction

Dans ce document, nous analyserons le framework JavaScript **VueJS** dans le but de comprendre le fonctionnement de la technologie. Ce document n'a pas pour but de faire du lecteur un expert VueJS, mais plutôt offrir les **connaissances élémentaires** du framework et apporter une **vision objective** et **globale** de la solution. De plus, les connaissances acquises à travers ce document nous permettront à mieux appréhender de telles technologies.

Pour ce faire, nous commencerons dans un premier temps par décortiquer les **concepts de bases** de VueJS afin de comprendre ce qu'offre le framework. Nous nous pencherons dans un second temps aux différentes **méthodes d'installation** du framework - en développement ainsi qu'en production. Puis, nous analyserons un **cas d'utilisation** afin de comprendre comment VueJS propose l'implémentation d'une fonctionnalité. Dans un quatrième temps nous allons **comparer** les points forts et points faibles de VueJS à ses concurrents actuels : Angular et React. Nous finirons le document par une **conclusion** de ce qui a été vu précédemment, afin d'en tirer un jugement.

Concepts de base

Comme la plupart de ses concurrents, VueJS a pour but de normaliser le fonctionnement d'une application web en mettant en place une organisation plus formelle et structurée. Pour ce faire, VueJS offre une architecture monolithique¹ dont le "core" du framework est centré sur la partie de **visualisation** d'une application. Le but premier de VueJS est en effet de visualiser des **données dynamiques**, tout en plaçant une partie de la logique d'affichage dans le code HTML.

```
<div id="app">
  {{ message }}
</div>
```

Figure 1

L'exemple ci-contre (fig. 1) est tout simplement une balise HTML dans laquelle vont être injectées des valeurs dynamiques. Pour ce faire, il suffit de créer une nouvelle **instance Vue** via la fonction **Vue** (fig. 2). L'objet passé en paramètre va permettre de définir le

comportement de l'application

Vue. Dans le cas présenté, la valeur "el" correspond à l'élément du DOM sur lequel l'instance Vue va être montée², tandis que la valeur "data" correspond à un set de valeurs sous la forme d'un objet JavaScript. Il serait également possible d'ajouter en paramètre une liste de méthodes s'exécutant suite à des actions effectuées par l'utilisateur, tel qu'un click, un hover ou autre.

```
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue!'
  }
})
```

Figure 2

Toute cette logique créée ce qui est appelé un **composant**. Chaque composant possède sa propre logique, ses propres éléments HTML, découpant ainsi l'application par fonctionnalité.

¹ https://en.wikipedia.org/wiki/Monolithic_application

² <https://vuejs.org/v2/api/#el>

Chaque instance de Vue possède un **cycle de vie** qui lui est propre. Chaque étape, tel que monter l'instance sur le DOM, mettre à jour le DOM ou encore interagir avec des données va appeler des méthodes dites “**lifecycle hook**”, offrant ainsi la possibilité de placer du code spécifique à une ou plusieurs étapes souhaitées.

L'exemple suivant (fig. 3) va par exemple appeler une méthode après l'instanciation de l'objet Vue. Le tout est comme précédemment passé en paramètre à la fonction Vue sous la forme d'un objet. Dans cet exemple, la **méthode** est “created”, désignant l'instant après l'instanciation. D'autres méthodes sont reliées à d'autres évènements tels que :

```
new Vue({
  data: {
    a: 1
  },
  created: function () {
    // `this` points to the vm instance
    console.log('a is: ' + this.a)
  }
})
// => "a is: 1"
```

Figure 3

- “Mounted” qui est appelé une fois que l'instance Vue est liée à un objet du DOM.
- “Updated” lors-ce que l'élément du DOM en question est modifié suite au changement d'une donnée.
- “Destroyed” lors ce que l'instance Vue est détruite.

Ses méthodes sont cependant effectuées **après** les **actions**. Par exemple le code de la méthode “updated” va être appelé après que les données ont été changées. Pour placer du code juste avant, il existe tout simplement les méthodes nommées “beforeCreate”, “beforeMount”, “beforeUpdate”, ou encore “beforeDestroy”.

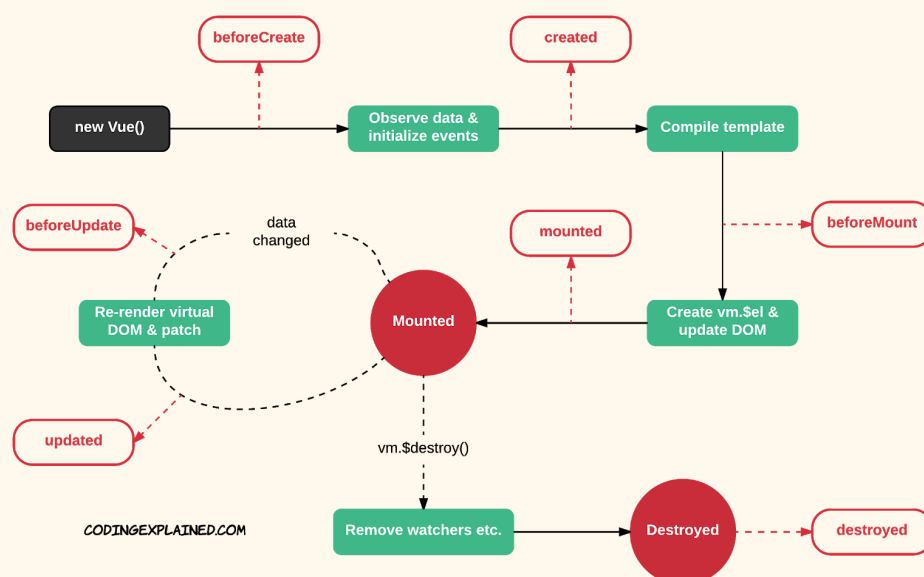


Figure 4

Les **templates** sont également des éléments clefs à la compréhension du framework, puisque VueJS va compiler les templates en éléments de DOM virtuel offrant ainsi la possibilité d’y injecter directement des données.

```
<span v-once>This will never change: {{ msg }}</span>
```

Figure 5

L'exemple ci-dessus (fig. 5) va afficher le contenu de “msg” de la propriété nommée “msg” de l’objet “data”. La **directive** “v-once” signifie que l’élément ne va être affiché, puis être considéré comme un élément de DOM statique.

```
<p v-if="seen">Now you see me</p>
```

Figure 6

L'exemple ci-dessus (fig. 6) va par exemple afficher le contenu de la balise en fonction de la valeur de “seen”. Il existe d’autres directives basées sur des **expressions Javascript**. Les directives VueJS sont préfixées par “v-” afin de les démarquer dans le template.

Afin de ne pas placer trop de logique dans les expressions de template, VueJS met en place des **computed properties** dont le but est de séparer l’affichage et la logique lors-ce que celui ci devient trop conséquent.

L'exemple ci-contre (fig. 7 & fig. 8) va par exemple permettre de créer une computed property nommée “reversedMessage” qui va s’occuper d’inverser les caractères de l’attribut “message” puis les retourner.

Le résultat aura donc la forme suivante :

Original message: “Hello”

Computed reversed message “olleH”

```
<div id="example">
  <p>Original message: "{{ message }}"</p>
  <p>Computed reversed message: "{{ reversedMessage }}"</p>
</div>
```


Figure 7

Figure 8

Procédure d'installation

VueJS peut être installé via 3 méthodes :


1. En téléchargeant et incluant manuellement les fichiers sources. Cette variante offre 2 types de sources : un mode de développement contenant les messages d'erreurs et d'avertissements, et un mode de production, plus compact et moins volumineux. Il est également possible d'inclure VueJS via un CDN (fig. 9)



```
<script src="https://cdn.jsdelivr.net/npm/vue@2.5.16/dist/vue.js"></script>
```

Figure 9

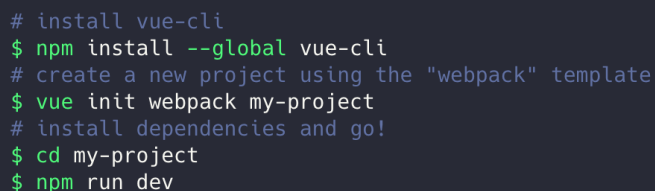
2. Via NPM, qui est la solution recommandée pour la mise en place de grande infrastructure (fig. 10).



```
# latest stable
$ npm install vue
```

Figure 10

3. Via le Command-line interface (CLI) officielle, qui permet de mettre en place en quelques commandes un “échafaudage” (une architecture) d'application. Le CLI offre également le **hot-reload** qui est l'actualisation de la page web lors-ce que les fichiers sources sont modifiés, du **lint-on-save** qui est une librairie d'analyse statique de code³, ou encore des **builds** destinés à la **production**. L'exemple ci-dessous (fig. 11) va installer le CLI ainsi que ses dépendances, créer un nouveau projet et lancer une exécution locale en mode développement.



```
# install vue-cli
$ npm install --global vue-cli
# create a new project using the "webpack" template
$ vue init webpack my-project
# install dependencies and go!
$ cd my-project
$ npm run dev
```

Figure 11

³ <https://www.jshint.com/>

Cas d'utilisation

L'exemple ci-dessous (fig. 12) réalisé en VueJS permet de **trier** et **filtrer** des **valeurs** dans un tableau dynamiquement.

Search

Name ▲	Power ▲
Chuck Norris	Infinity
Bruce Lee	9000
Jackie Chan	7000
Jet Li	8000

Figure 12

Pour ce faire, le code suivant est utilisé (fig. 13). Le contenu des méthodes de filtres, de mise en majuscule et de tri a été coupé afin d'améliorer la compréhension du code.

Nous retrouvons donc le **template** nommé "grid-template" qui va itérer sur le set de données et monter les méthodes de tri et de filtre sur les bons éléments du DOM.

Plus d'exemples d'utilisations sont présentés dans la documentation officielle VueJS⁴.

```
// register the grid component
Vue.component('demo-grid', {
  template: '#grid-template',
  props: {
    data: Array,
    columns: Array,
    filterKey: String
  },
  data: function () {
    var sortOrders = {}
    this.columns.forEach(function (key) {
      sortOrders[key] = 1
    })
    return {
      sortKey: '',
      sortOrders: sortOrders
    }
  },
  computed: {
    filteredData: function () {
      return ...
    }
  },
  filters: {
    capitalize: function (str) {
      return ...
    }
  },
  methods: {
    sortBy: function (key) {
      ...
    }
  }
})

// bootstrap the demo
var demo = new Vue({
  el: '#demo',
  data: {
    searchQuery: '',
    gridColumns: ['name', 'power'],
    gridData: [
      { name: 'Chuck Norris', power: Infinity },
      ...
    ]
  }
})
```

Figure 13

⁴<https://vuejs.org/v2/examples/>

Comparaison

Maintenant que nous avons une vision globale de ce que peut offrir VueJS, nous allons comparer le framework avec ses 2 grands concurrents : Angular et React.

Maturité.

Google qui a créé **Angular** l'a rendu disponible au grand public en 2010. Cependant, certains bugs sont toujours actifs sur le framework. Angular 2, qui se veut plus stable, a été introduit en 2016. Angular est notamment utilisé par Google, Wix ou encore weather.com.

React date de 2013 est qui est développé et maintenu par **Facebook**. React est utilisé par Uber, Netflix, Twitter, Airbnb, Reddit ou encore Stripe.

Contrairement à ses deux concurrents, **Vue** n'a pas été développé par une entreprise, mais par un ex-employé de Google ainsi qu'une grande communauté. C'est en 2014 qu'a vu le jour Vue, et est désormais utilisé par Alibaba, Nintendo ou encore Gitlab.

Popularité.

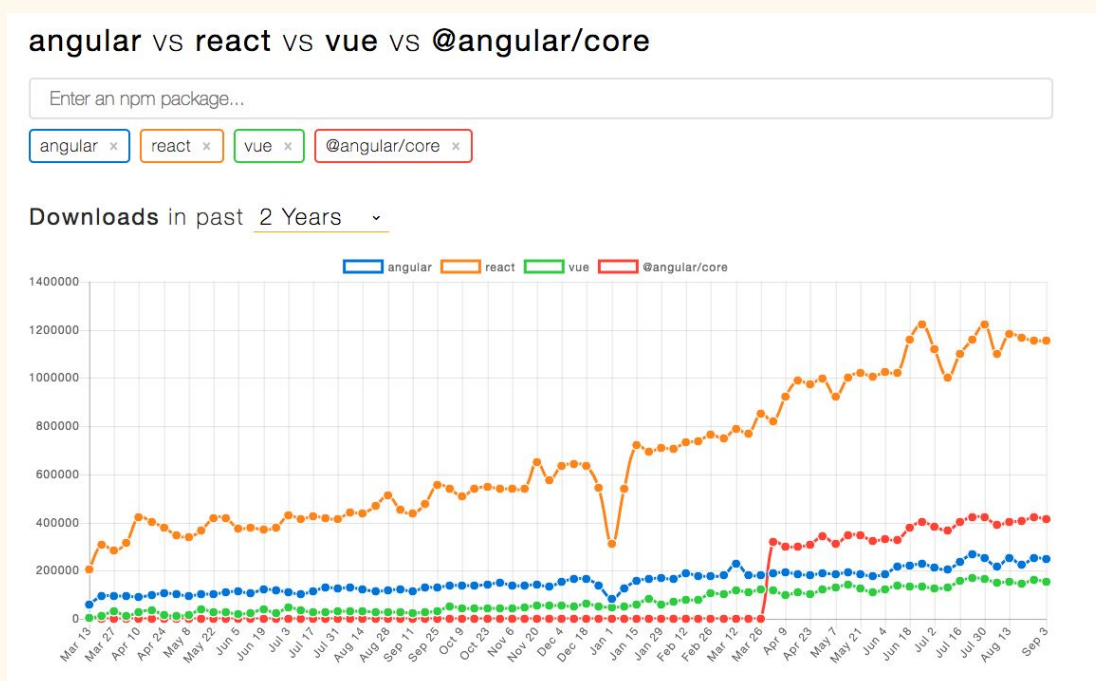


Figure 14

Le graphique ci-dessus (fig. 14) montre les **tendances npm** de ces deux dernières années⁵.

⁵ <http://www.npmtrends.com/angular-vs-react-vs-vue-vs-@angular/core>

Flexibilité.

Angular est plus un framework qu'une librairie, contraignant ainsi les libertés de structure de l'application, et possède plus de fonctionnalités dites **"out of the box"**.

Par opposition, React et Vue se veulent être plus flexibles et peuvent s'imbriquer plus facilement à divers modules.

Performances.

Le tableau ci-contre (fig. 15) indique que Vue a de meilleures performances face à ses concurrents. Cependant il semblerait que les tests de performance de telles technologies sont à facilement manipulable⁶.

Apprentissage.

Angular semble être le framework le plus complexe à prendre en main, et ce même si la documentation est complète. De plus, Angular utilise Typescript⁷, ajoutant ainsi une couche supplémentaire de technologie à connaître.

Vue reste l'option la plus facile à prendre en main. Par exemple, React nécessite par exemple de choisir un **gestionnaire d'état**⁸ parmi de nombreuses solutions. Avec Vue, la barrière séparant les développeurs juniors et seniors rétrécit grandement, menant à une meilleure collaboration et compréhension de code.

Name	angular-v4.1.2-keyed	react-v15.5.4-redux-v3.6.0	vue-v2.3.3-keyed
create rows Duration for creating 1000 rows after the page loaded.	193.1 ± 7.9 (1.2)	212.2 ± 14.2 (1.3)	166.7 ± 8.8 (1.0)
replace all rows Duration for updating all 1000 rows of the table (with 5 warmup iterations).	197.4 ± 5.3 (1.2)	206.7 ± 7.3 (1.2)	168.5 ± 5.0 (1.0)
partial update Time to update the text of every 10th row (with 5 warmup iterations).	13.0 ± 4.5 (1.0)	18.0 ± 1.6 (1.1)	17.3 ± 2.9 (1.1)
select row Duration to highlight a row in response to a click on the row. (with 5 warmup iterations).	3.4 ± 2.3 (1.0)	8.7 ± 2.9 (1.0)	9.3 ± 1.7 (1.0)
swap rows Time to swap 2 rows on a 1K table. (with 5 warmup iterations).	13.4 ± 1.0 (1.0)	17.1 ± 1.3 (1.1)	18.3 ± 1.5 (1.1)
remove row Duration to remove a row. (with 5 warmup iterations).	46.1 ± 3.2 (1.0)	52.4 ± 1.7 (1.1)	52.6 ± 2.7 (1.1)
create many rows Duration to create 10,000 rows	1946.0 ± 41.8 (1.2)	1931.7 ± 35.6 (1.2)	1587.5 ± 33.9 (1.0)
append rows to large table Duration for adding 1000 rows on a table of 10,000 rows.	324.6 ± 10.1 (1.0)	366.4 ± 10.9 (1.1)	399.5 ± 11.0 (1.2)
clear rows Duration to clear the table filled with 10,000 rows.	379.9 ± 11.3 (1.5)	410.9 ± 9.8 (1.6)	254.5 ± 5.0 (1.0)
startup time Time for loading, parsing and starting up	84.3 ± 2.6 (1.5)	93.8 ± 6.9 (1.7)	56.6 ± 2.5 (1.0)
slowdown geometric mean	1.14	1.23	1.06

Figure 15

⁶ <https://medium.com/@localvoid/how-to-win-in-web-framework-benchmarks-8bc31af76ce7>

⁷ <https://www.typescriptlang.org/docs/handbook/angular.html>

⁸ <https://github.com/voronianski/flux-comparison>

Conclusion

Le fait que Vue est développé/maintenu par une communauté et non une multinationale est à la fois un avantage et un désavantage. Vue est financé grâce à des donations, ce qui restreint le nombre de personnes travaillant entièrement sur le projet. Cependant, le fait d'avoir une équipe plus modeste a tendance à empêcher l'“over engineering” de la technologie.

Vue offre en effet beaucoup de libertés quant à l'implémentation d'une solution. Contrairement à ses concurrents, Vue vient avec moins d'**outils** en tous genres qui mènent souvent à des confusions. Vue semble être un bon compromis en termes de simplicité lié aux dépendances.

Mon avis personnel est le suivant : Vue est une technologie front-end “nice-to-know”. Même si Vue est bien moins populaire que React, il s'agit d'une excellente alternative “plus simple et plus épurée”. L'écosystème Javascript devient omniprésent sur le web, menant à une fatigue⁹ lié à la multitude de possibilités et d'outils disponibles. Vue offre pour sa part une base de connaissances saine avec une décente courbe d'apprentissage.



⁹ <https://medium.com/@ericclemmons/javascript-fatigue-48d4011b6fc4>