

Brice Chaponneau

Vue.js

Applications web complexes et réactives



● Éditions
EYROLLES

Le framework JavaScript concurrent de React et Angular



Un ouvrage de référence pour le développeur web

Vue.js est un framework JavaScript orienté front-end qui mérite considération à plusieurs égards. Il est réactif, performant, versatile, facilement testable, maintenable et sa courbe d'apprentissage est réellement rapide.

L'écriture globale est idéalement structurée et son écosystème aide à créer, organiser et maintenir vos applications clientes.

Ce framework peut se suffire à lui-même pour développer des applications complexes en ayant recours à de simples composants, des mixins ou des plug-ins. De plus, il s'accompagne d'un univers où de multiples outils sont disponibles pour aider au développement : des extensions, des plug-ins et des bibliothèques complètes pour vous faire gagner en temps de réalisation, en qualité de code et également en performance.

Compléments web

Tous les exemples des programmes du livre sont en téléchargement sur notre site Internet : www.editions-eyrolles.com/dl/0067783.

À qui s'adresse cet ouvrage ?

- Aux développeurs et chefs de projet web qui souhaitent réaliser des applications web performantes.
- À toutes les personnes qui souhaitent découvrir Vue.js et acquérir des connaissances certaines afin d'être autonomes dans le développement web autour de ce framework.

Au sommaire

Installer et utiliser Vue.js • Les outils préconisés et leur configuration • Les paradigmes fondamentaux de Vue.js • Les directives pour commander les éléments • Les directives personnalisées • Formater avec l'interpolation des filtres • Les composants • Les slots, un emplacement réservé pour injecter du contenu • Le composant keep-alive pour garder l'état courant • Apporter de la dynamique visuelle avec les transitions • La réutilisabilité avec les mixins • Ajouter des fonctionnalités avec les plug-ins • Extension de composant • Le store, gestionnaire d'états • API • Le routage pour la navigation

Brice Chaponneau est titulaire d'un Master IT dans le développement des applications réparties. Il a travaillé dans différents domaines dont la banque (Caisse d'Épargne, Société Générale, Edmond de Rothschild, Natixis), l'assurance (Monceau Assurance), les transports (SNCF) et le secteur industriel (ArcelorMittal). Brice a donc été développeur, lead technique, chef de projet MOE, Scrum Master et consultant. Il a majoritairement développé en JavaScript, .Net et via des frameworks divers. Il maîtrise les bases de données NoSQL comme Mongo DB et SGBDR (SQL Server, Oracle, Sybase).

www.editions-eyrolles.com

Vue.js

DANS LA MÊME COLLECTION

- S. RINGUEDÉ. – **SAS.**
N° 67631, 2019, 688 pages.
- M. BIDAULT. – **Programmation Excel avec VBA.**
N° 67786, 2019, 512 pages.
- R. GOETTER. – **CSS 3 Grid Layout.**
N° 67683, 2019, 131 pages.
- C. BLAESS. – **Solutions temps réel sous Linux.**
N° 67711, 3^e édition, 2019, 318 pages.
- C. PIERRE DE GEYER, J. PAULI, P. MARTIN, E. DASPET. – **PHP 7 avancé.**
N° 67720, 2^e édition, 2018, 736 pages.
- H. WICKHAM, G. GROLEMUND. – **R pour les data sciences.**
N° 67571, 2018, 496 pages.
- F. PROVOST, T. FAWCETT. – **Data science pour l'entreprise.**
N° 67570, 2018, 370 pages.
- J. CHOKOGOUE. – **Maîtrisez l'utilisation des technologies Hadoop.**
N° 67478, 2018, 432 pages.
- H. BEN REBAH, B. MARIAT. – **API HTML 5 : maîtrisez le Web moderne !**
N° 67554, 2018, 294 pages.
- W. MCKINNEY. – **Analyse de données en Python.**
N° 14109, 2015, 488 pages.
- E. BIERNAT, M. LUTZ. – **Data science : fondamentaux et études de cas.**
N° 14243, 2015, 312 pages.

SUR LE MÊME THÈME

- É. SARRION. – **React.js.**
N° 67756, 2019, 350 pages.
- T. PARISOT. – **Node.js.**
N° 13993, 2018, 472 pages.
- C. HERBY. – **Apprenez à programmer en Java.**
N° 67521, 2018, 788 pages.

Retrouvez nos bundles (livres papier + e-book) et livres numériques sur
<http://izibook.eyrolles.com>

Brice Chaponneau

Vue.js

Applications web complexes et réactives

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

En application de la loi du 11 mars 1957, il est interdit de reproduire intégralement ou partiellement le présent ouvrage, sur quelque support que ce soit, sans l'autorisation de l'Éditeur ou du Centre Français d'exploitation du droit de copie, 20, rue des Grands Augustins, 75006 Paris.

© Éditions Eyrolles, 2019, ISBN : 978-2-212-67783-6

Avant-propos

Comment lire cet ouvrage ?

L'ouvrage est construit de manière incrémentale sur la compréhension et l'utilisation de Vue.js : de l'installation du cœur et des packages principaux de ce framework à la création d'un projet complet, en passant par l'ensemble de ses composantes, la préconisation d'outils et leur configuration mais également des astuces d'écriture de code ou d'optimisation d'exécution.

Que contient ce livre ?

Pour appuyer les chapitres décrivant une composante majeure de Vue, des exercices seront proposés afin de bien en comprendre les rouages. Ils mèneront à un projet final. Il est donc conseillé de lire ce livre chapitre après chapitre.

Le premier chapitre est la mise en lumière de Vue. Le deuxième chapitre expose la multitude de possibilités offertes pour installer et/ou utiliser Vue et propose un panel d'outils, ainsi que leur configuration. Les chapitres suivants décortiquent les paradigmes du framework.

À qui s'adresse-t-il ?

Vue est écrit en JavaScript (JS) et ce livre n'a pas vocation à reprendre les bases de ce langage, ni les conventions ECMAScript – même si nous y trouverons quelques rappels et comparaisons – mais à disséquer ce framework. Il est donc nécessaire que le lecteur connaisse les bases de JS, du langage HTML et des styles CSS. Notons cependant que les exemples vont à l'essentiel afin d'exposer un sujet précis et ne veulent en aucun cas perdre le lecteur dans des propositions surchargées.

Code source en téléchargement

Tous les codes sources sont également disponibles sur l'espace de téléchargement dédié sur le site des éditions Eyrolles (editions-eyrolles.com/dl/0067783) afin de permettre au lecteur de se focaliser sur la compréhension et les tests plutôt que sur la réécriture.

Un fichier nommé `Readme.html` à la racine de ce dossier de téléchargement apporte une explication simple et détaillée pour pouvoir lancer les serveurs et apprécier le code ainsi que le rendu.

Table des matières

Introduction	1
Historique de Vue.js	1
Comparatif des frameworks JavaScript actuels	1
Rappels de modélisation en génie logiciel	5
Architecture MVC	5
Architecture MVVM	5
Front-end et back-end	6
 CHAPITRE 1	
Installer et utiliser Vue.js	7
Une version par environnement	7
Sources du framework	8
Autonome – Source officielle	8
CDN – Serveur de distribution	8
Nuxt – Le framework universel	8
Vue CLI – Le générateur de projet officiel	8
CodeSandbox – Une solution clé en main	9
Bower – Un gestionnaire de dépendances	9
NPM (ou Yarn) – Le gestionnaire de référence	9
 CHAPITRE 2	
Les outils préconisés et leur configuration	13
VS Code Debugger for Chrome	13
Description	13
Configuration	13
Déboguer pas à pas	15

Vue.js devtools	16
Description	16
Installation	17
Vue Performance Devtool	17
Description	17
Vetur	18
Description	18

CHAPITRE 3

Les paradigmes fondamentaux de Vue.js	19
Instance de Vue.js	19
Au cœur du système réactif	19
Organisation et optimisation de la structure de page HTML	21
Cycle de vie d'une instance de Vue.js	22
Hooks du cycle de vie	23
Le contexte ou la portée	23
Hello World – Première instance	24
Qu'est-ce qu'un composant et comment l'intégrer ?	25
Structure d'un composant	25
Intégration d'un composant dans l'environnement applicatif	25
Les propriétés d'instance	26
Référencer les éléments avec \$refs	26
Interpolation pour générer du rendu	29
Texte ou mustache	30
Utilisation des premières directives	31

CHAPITRE 4

Les directives pour commander les éléments	35
Associer les directives et les arguments	36
Des modificateurs pour enrichir les directives	37
Les directives natives en détail	37
Les directives d'interpolation	37
Les directives de rendu conditionnel	39

Les directives de rendu de liste	41
Les directives de gestion d'événements	48
Les directives de liaison	53
 CHAPITRE 5	
Les directives personnalisées	65
Enregistrement et utilisation	65
Exemple : directive de déplacement	67
 CHAPITRE 6	
Formater avec l'interpolation des filtres	71
Qu'est-ce qu'un filtre ?	71
Écriture de filtres	71
 CHAPITRE 7	
Les composants	75
Définition	75
Premier composant	76
Les options structurantes	78
Data : les variables réactives	78
Props : les propriétés de communication	79
Methods : les fonctions de traitement	83
Computed : le calcul avec mise en cache	85
Différences entre les options methods et computed	87
Watch : personnaliser l'observation des changements	87
V-model personnalisé	90
Création locale	93
Création globale	94
Création mono-fichier	94
Écriture alternative	96
Optimisation avec l'architecture modulaire	97
Optimisation avec le lazy loading	99
Préconisation pour écrire rapidement un composant	100

Communiquer avec les composants	101
Communication parent vers enfant	103
Communication enfant vers parent	103
Communication entre composants	105
Communication avec un composant récursif	107
Gestion de composants dynamiques	111
Supprimer l'héritage d'attribut	114
 CHAPITRE 8	
Les slots, un emplacement réservé pour injecter du contenu	117
Définition	117
Utilisation standard d'un slot	118
Utilisation de slots nommés	121
Slot avec portée	123
Slot avec passage de propriété	124
Des slots dynamiques	125
Comment et pourquoi tester l'existence d'un slot ?	126
 CHAPITRE 9	
Le composant keep-alive pour garder l'état courant	131
Utilisation du système de cache	131
 CHAPITRE 10	
Apporter de la dynamique visuelle avec les transitions	135
Qu'est-ce que le composant transition ?	135
Mémento des classes et des événements pour les transitions	137
Exemple de transition	137
Des transitions réutilisables et génériques	140

CHAPITRE 11

La réutilisabilité avec les mixins	147
Qu'est-ce qu'un mixin ?	147
Attentions à porter lors de l'utilisation des mixins	148

CHAPITRE 12

Ajouter des fonctionnalités avec les plug-ins	153
Comment créer un plug-in ?	154
Plug-in d'intégration	155
Installation automatique	155
Comment utiliser un plug-in ?	156

CHAPITRE 13

Extension de composant	159
La méthode	159
L'injection de dépendances	163
Principe	163

CHAPITRE 14

Le store, gestionnaire d'états	167
Définition	167
Le gestionnaire Vuex	170
Qu'est-ce que Vuex ?	170
Vue devtools comme compagnon	171
Comment installer Vuex ?	171
Création de la structure de base	172
Créer un store avec Vuex	173

CHAPITRE 15

API	201
Principe de base de communication avec une API	201
Comment communiquer avec une API ?	201
Bibliothèque Fetch	201
Bibliothèque Axios	203
Consommation d'une API	206

CHAPITRE 16

Le routage pour la navigation	217
Pourquoi utiliser un plug-in de routage ?	217
Comment installer le router ?	218
Comment définir une route ?	218
Préconisation pour la gestion du routage	219
Un composant pour page	222
La concordance dynamique, mettre des variables dans nos routes	225
La vue, naviguer sur une page avec de multiples composants	229
Naviguer par le code, sans action de l'utilisateur	231
Intercepter une route de navigation	232
De la métadonnée dans les routes	234
De l'animation dans la navigation	234
Conclusion	239
Index	241

Introduction

Vue (prononcé « view ») est un framework évolutif JavaScript front-end et open source qui permet de construire des interfaces web utilisant des liaisons de données MVVM (Modèle-Vue-Vue-Modèle) très simplement, le tout architecturé autour du composant et surtout de la réutilisabilité.

Il est possible de réaliser des composants unitaires, des applications SPA (*Single Page Application*), SSR (*Server Side Rendering*), Mobile... De plus, un projet Vue peut être couplé avec d'autres outils ou bibliothèques tierces.

Vue est disponible sur le site officiel du framework à l'adresse suivante : <https://vuejs.org/>.

Historique de Vue.js

Après avoir travaillé chez Google sur divers projets avec leur framework AngularJS, Evan You a implémenté un framework plus léger, organisé et plus modulable. La première version de Vue a été déposée sur GitHub en février 2014 et son code couvert par des tests unitaires sous Karma (bibliothèque JavaScript de tests unitaires).

Aujourd'hui, ce projet est maintenu par divers auteurs à l'international tant pour le noyau que pour les outils et modules complémentaires.

Comparatif des frameworks JavaScript actuels

À ce jour, il existe une multitude de bibliothèques JavaScript telles que JQuery, Angular, Ember et React pour ne citer que les plus populaires. Voici quelques comparatifs que nous pouvons apprécier concernant l'évolution de l'utilisation et de la cote de popularité de Vue.js et ce, malgré sa jeunesse somme toute relative.

La figure I-1 présente les statistiques NPM (graphique) et GitHub (tableau) des frameworks JS sur 2 ans, avec comme métriques la popularité et la tendance d'utilisation avec les curseurs suivants :

- *stars* : popularité des utilisateurs ;
- *forks* : nombre de copies des référentiels faites par les utilisateurs ;
- *issues* : nombre d'anomalies remontées par les utilisateurs.

On observe donc qu'à ce jour, React.js est en première position et Vue.js en deuxième position. Il faut garder en mémoire que React a eu la promotion de Facebook et que ce framework est né 1 an avant Vue. Evan You a su promouvoir son projet, le rendre fiable et populaire.

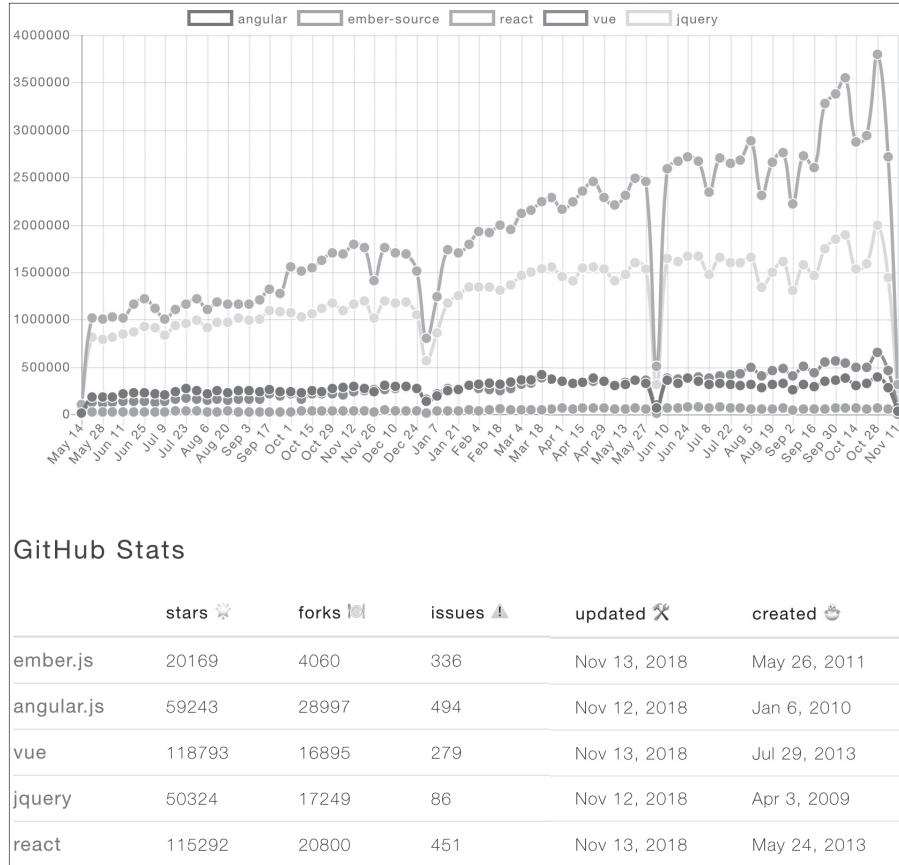


Figure I-1 – Comparatif des frameworks JS
(Source : <https://www.npmtrends.com>)

Cet autre site donne d'abord les statistiques de popularité (onglet *Popular*) et dans un second temps la tendance d'utilisation (onglet *Trending*) des frameworks actuels (figures I-2 et I-3).

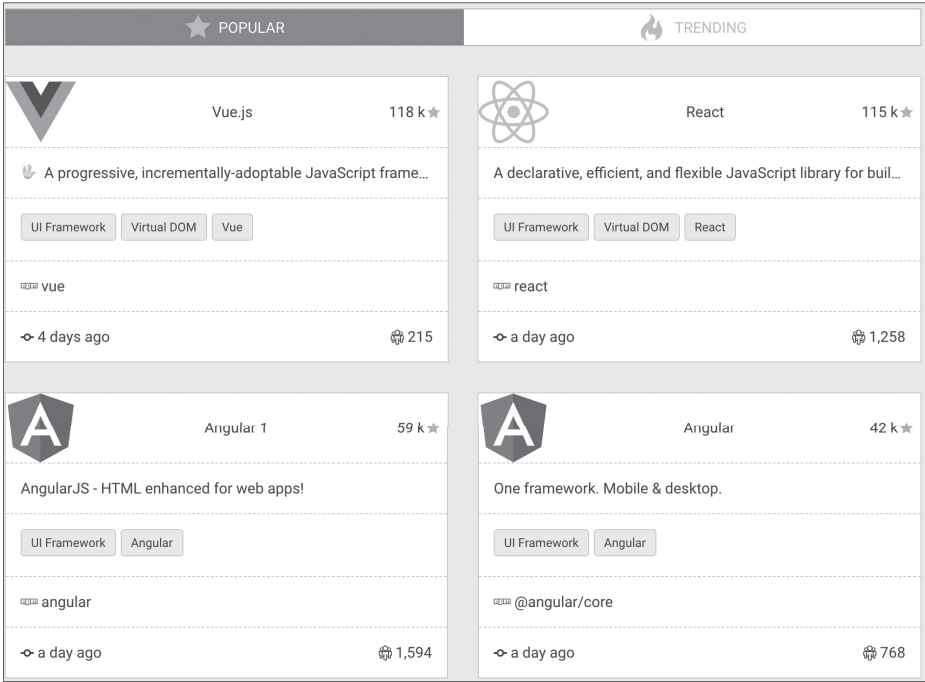


Figure I-2 – Popularité des frameworks JS sur 1 an
(Source : <https://bestof.js.org>)

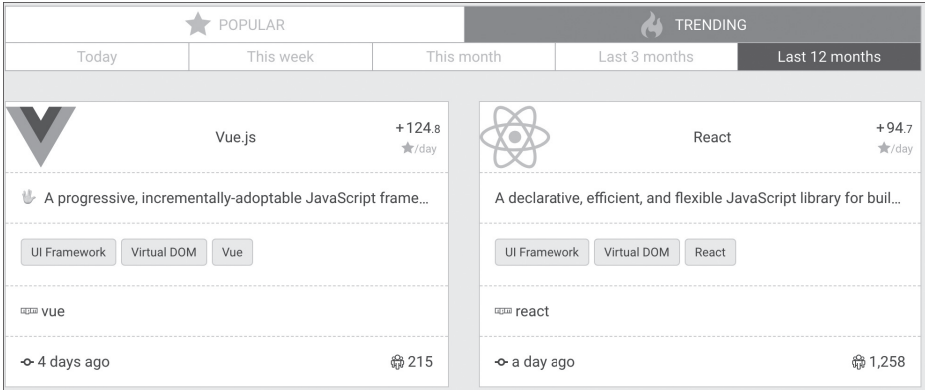


Figure I-3 – Tendance de l'utilisation des frameworks JS sur 1 an
(Source : <https://bestof.js.org>)

Pour finir, observons un benchmark lancé sur un MacBook Pro 15 (2,5 GHz i7, 16 Go RAM, OS X 10.12.5, Chrome 58.0.3029.110, 64-bit) dans lequel nous avons pour le premier tableau le

temps d'exécution (exprimé en millisecondes) de plusieurs méthodes classiques, puis dans le second tableau l'allocation mémoire utilisée après le chargement de la page et après avoir ajouté 1 000 lignes (figure I-4).

Plus la cellule du tableau tend vers le foncé, pire est le traitement, et inversement lorsqu'elle tend vers le clair.

Nous constatons que le VanillaJS (nom moderne pour parler de JavaScript natif) est forcément en première position et que Vue est en deuxième position !

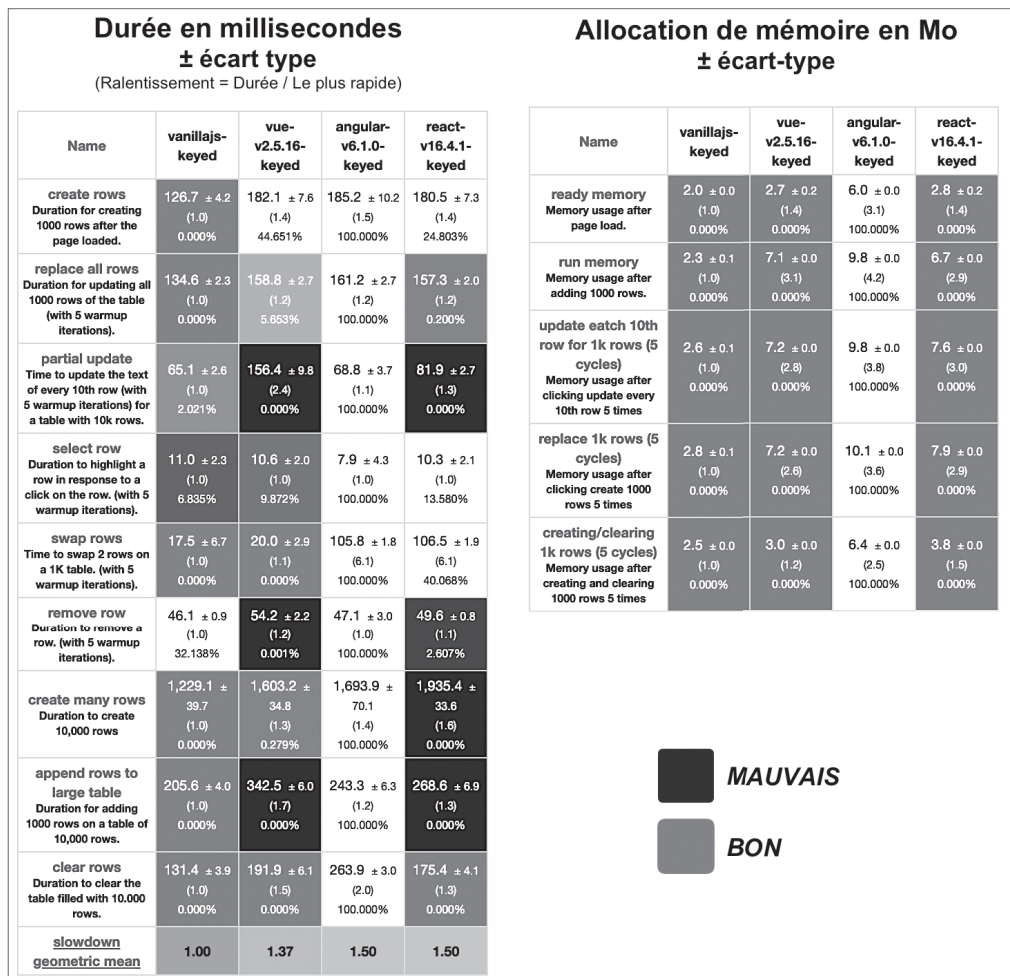


Figure I-4 – Benchmark de rapidité d'exécution et d'allocation mémoire des frameworks JS
(Source : <https://www.stefankrause.net>)

Rappels de modélisation en génie logiciel

Architecture MVC

En génie logiciel, le modèle-vue-contrôleur (*Model-view-controller*) est une architecture destinée à découper une application en couches (figure I-5), surtout pour les interfaces web.

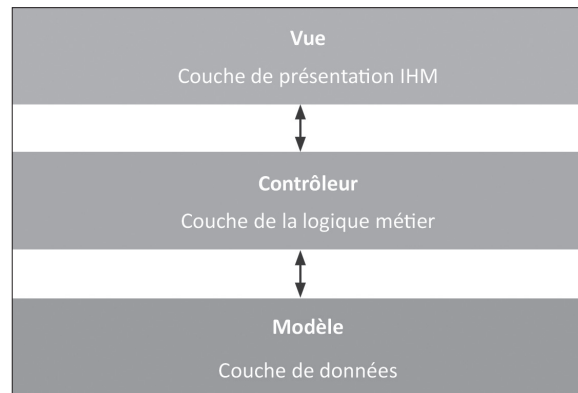


Figure I-5 – Schéma du modèle MVC

Architecture MVVM

En génie logiciel, le modèle-vue-vue modèle (*Model-view-viewmodel*) est une architecture et une méthode de conception qui est originaire de Microsoft, notamment pour WPF et Silverlight. Très similaire au modèle MVC avec une accentuation des principes de binding (liaison) et events (événements). C'est sur cette modélisation que s'appuie le framework Vue.js.

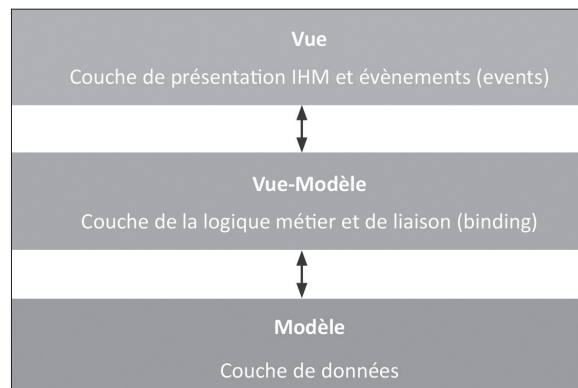


Figure I-6 – Schéma du modèle MVVM

Front-end et back-end

Ici, nous parlons d'organisation, de rôle, de métier...

Le back-end est la partie « immergée de l'iceberg », invisible pour les utilisateurs. Il est le cœur de l'application où l'on retrouve les données et le cœur métier.

Le front-end, quant à lui, est la « pointe de l'iceberg », visible par les utilisateurs. Il représente l'interface : les formulaires et le design de l'application.

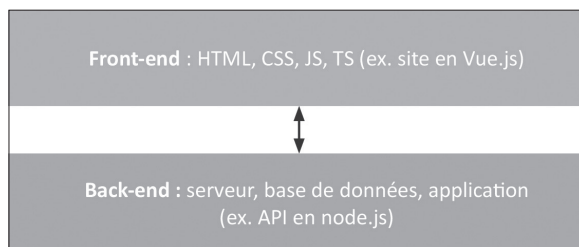


Figure I-7 – Schéma de communication front-end et back-end

1

Installer et utiliser Vue.js

Une version par environnement

Il est possible d'utiliser Vue de différentes manières, mais gardons en tête qu'il existe deux packages très distincts :

- la version de développement : débogage facilité et interaction avec divers outils (voir chapitre 2 sur les outils, page 13) ;
- la version de production : compressée au maximum (minifiée), ce qui aura pour désavantage de ne pas disposer de la richesse des messages d'informations du framework.

Important

Vue ne supporte pas les versions d'Internet Explorer (IE) 8 et inférieures car il utilise des fonctionnalités d'ECMAScript 5.

IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android	IE Mobile	UC Browser for Android
6-7								2.1-3						
8		2-3.6	4-18		10-11.5			4						
9		4-20	19-22	3.1-5.1	12.1	3.2-5.1		4.1-4.3		12				
10	12-16	21-62	23-69	6-11.1	15-55	6-11.4		4.4-4.4.4	7	12.1			10	
11	17	63	70	12	56	12	all	67	10	46	69	62	11	11.8
	18	64-65	71-73	TP										

Figure 1-1 – Compatibilité des navigateurs pour ES5
(Source : <https://caniuse.com/#feat=es5>)

Sources du framework

Autonome – Source officielle

La version Autonome signifie que nous avons en notre possession les sources de Vue. Il suffit de télécharger le fichier désiré et de l'inclure avec un tag script dans son fichier HTML. Ensuite, Vue doit être enregistré comme une variable globale (voir chapitre 3 : Instance de Vue) :

- développement : <https://fr.vuejs.org/js/vue.js>
- production : <https://fr.vuejs.org/js/vue.min.js>

CDN – Serveur de distribution

Un CDN (*Content Delivery Network*) stocke sur ses serveurs les sources du framework et nous les propose via un lien directement exploitable dans nos sources. Le site officiel de Vue.js préconise d'utiliser le CDN de `unpkg` qui fournit la dernière version stable possible afin de refléter le package fournit par NPM. Mais il est possible de s'appuyer sur d'autres CDN tels que `jsdelivr` et `cdnjs`.

- <https://unpkg.com/vue@2.5.17/dist/vue.min.js>
- <https://cdn.jsdelivr.net/vue/2.3.2/vue.min.js>
- <https://cdnjs.cloudflare.com/ajax/libs/vue/2.3.4/vue.min.js>

Note

Par défaut, la version proposée est la version minifiée, mais il est possible de supprimer la mention `.min` dans l'URL afin d'obtenir la version de développement.

Nuxt – Le framework universel

Nuxt est un projet dédié à Vue.js qui embarque en son sein les packages de Vue, Webpack et Babel (*transpiler* qui convertit le code ES en JS). Sont intégrés dans le package `vue-router`, `Vuex` et `vue-meta`. Il suffit de se rendre sur le site <https://fr.nuxtjs.org/> pour en savoir plus.

Vue CLI – Le générateur de projet officiel

Vue.js offre une CLI (*Command Line Interface*) officielle qui est une interface en ligne de commande pour générer un projet configuré avec les dépendances préconisées. Dans le terminal, il convient de saisir la commande pour l'installation de la CLI :

Installation de la CLI

```
| npm install -g @vue/cli
```

Puis de créer un projet :

Création du projet

```
| vue create nom-de-mon-projet
```

Il faut ensuite se rendre sur la page <https://cli.vuejs.org/> pour lire la documentation officielle détaillée.

CodeSandbox – Une solution clé en main

Le site CodeSandbox (<https://codesandbox.io/s/vue>) fournit un environnement complet et personnalisable pour développer rapidement et sans aucune installation requise (intégration de VS Code). Il peut être, et c'est d'ailleurs recommandé, couplé avec un compte Git.

CodeSandbox est aussi capable d'injecter des dépendances tierces. C'est un véritable IDE (*Integrated Development Environment* ou Environnement de développement) complet en ligne qui, par ailleurs, propose VS Code en guise d'outil d'écriture de code.

Bower – Un gestionnaire de dépendances

Bower est un outil, un gestionnaire de dépendances, à installer sur sa machine. C'est une solution que nous ne développerons pas ici mais qui mérite d'être mentionnée. Pour ceux qui l'utilisent, saisissez la ligne suivante dans l'invite de commande :

```
| bower install vue
```

NPM (ou Yarn) – Le gestionnaire de référence

NPM est l'outil indispensable à avoir sur sa machine. Il est installé avec Node.js et est disponible à l'adresse suivante : <https://www.npmjs.com/get-npm>.

Utilisé avec le terminal, il permet de gérer les dépendances et s'intègre très facilement avec un empaqueteur de modules (*module bundler*) tel que Webpack, Parcel, Rollup...

Après l'avoir installé, il convient de saisir la ligne suivante dans l'invite de commande :

```
| npm install vue
```

Empaqueurs de modules

Voici les configurations pour les empaqueteurs les plus populaires et un benchmark comparatif.

Webpack

Disponible sur la page <https://webpack.js.org/>, Webpack est l'empaqueur le plus populaire. Il demande néanmoins un temps assez important de compréhension et surtout de configuration avant de pouvoir monter un projet. Voici sa configuration pour Vue :

```
module.exports = {  
  // ...  
  resolve: {  
    alias: {  
      'vue$': 'vue/dist/vue.esm.js'  
    }  
  }  
}
```

Rollup.js

Rollup est assez proche de Webpack en termes de configuration. Cependant, c'est celui qui met le moins de temps à compiler les sources, que ce soit pour le mode Développement comme pour le mode Production. Il est disponible à l'adresse <https://rollups.org/guide/en> et sa configuration (fichier `package.json`) pour Vue est la suivante :

```
const alias = require('rollup-plugin-alias')  
  
rollup({  
  // ...  
  plugins: [  
    alias({  
      'vue': 'vue/dist/vue.esm.js'  
    })  
  ]  
})
```

Parcel.js

Parcel est le plus léger des empaqueteurs, tant en poids qu'en configuration. Il a pour vocation de ne nécessiter aucune configuration, tout est intégré. Il est sûrement à privilégier pour de petits projets et des POC (*proof of concept*). Pour plus d'information sur Parcel, consulter le site <https://parceljs.org/>. Pour ajouter la prise en charge des fichiers `.vue`, il suffit d'ajouter le code suivant dans le fichier `package.json` :

```
{  
  // ...  
  "alias": {  
    "vue": "../node_modules/vue/dist/vue.common.js"  
  }  
}
```

Benchmark

Pour un même projet, relancé trois fois, voici les temps d'exécution en secondes pour chacun des empaqueteurs :

Tableau 1-1. Benchmark d'exécutions des empaqueteurs

Empaqueteur	1 ^{er} lancement	2 ^e lancement	3 ^e lancement	Moyenne
Webpack	3,828	3,456	3,902	3,728
Rollup.js	0,650	0,498	0,495	0,547
Parcel.js	15,05	5,674	4,876	8,533

Notons que ce test est soit pour le mode Développement, soit pour le mode Débogage. Pour le mode Production, les temps sont sensiblement les mêmes sauf pour Parcel.js qui est complètement hors-jeu avec un temps de premier lancement multiplié par dix.