MIDI TECH

Vue.js, une alternative à Angular et React

François Delbrayelle (@fdelbrayelle)

BIENVENUE !

- Midis techniques S03E03
- Vous aussi vous pouvez en faire ;-) !
- Présentation puis démo
- **Sondage : qui connaît/utilise Vue.js ?**

Vue.js

QU'EST-CE QUE C'EST ?

- Un framework front JS (ES2015/2016)
- Concurrent à Angular et React
- Sans lien avec un GAFAM
- Créé par Evan You en 2014
- Open-source, licence MIT
- Version actuelle : 2.6

CARACTÉRISTIQUES

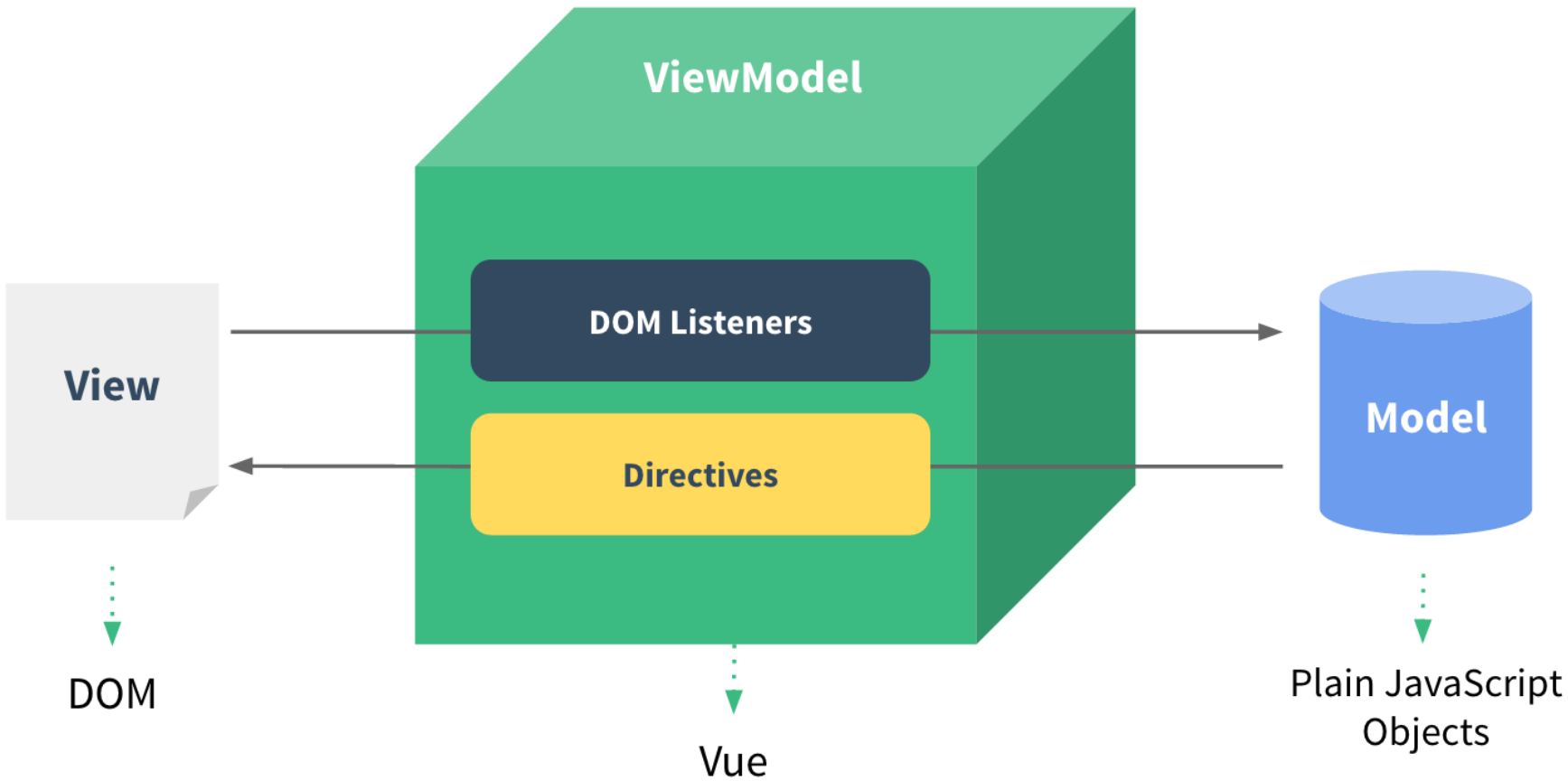
- Basé sur les composants/vues
- Architecture proche du MVVM
- Virtual DOM comme React (compile les templates en fonctions de rendu)
- Vue Router, VueX (State)...
- Apprentissage rapide et facile
- Possibilité d'utiliser TypeScript

			
	A declarative, efficient, and flexible JavaScript library for building user interfaces.	One framework. Mobile & desktop.	A progressive, incrementally adoptable JavaScript framework for building UI on the web.
	116 993 ⭐	59 302 ⭐	121 050 ⭐
Original author	Jordan Walke	Miško Hevery	Evan You
Developers	Facebook	Google	
Initial release	May 29, 2013	October 20, 2010	February 2014
Npm weekly downloads	3 940 035	433 361	709 943
Size	109.7 KiB production 774.7 KiB development	167 kB production 1.2 MB development	30.67 KB production 279 KB development
Easy to learn	Medium	Learn TypeScript	Yes
Coding speed	Normal	Slow	Fast
Documentation	✓	✓	✓
Performance	✓	✓	✓
Startup time	Quick	Longer due to its large codebase	Quick
Complete web apps	Needs to be integrated with many other tools	Can be used on standalone basis	Requires third party tools
Data binding	Uni-directional	Bi-directionall	Bi-directional
Rendering	Server side	Client side	Server side

INSTALLATION

- En inclusion directe `script`
- Avec `npm install vue` : Webpack...
- Avec Vue CLI (cli.vuejs.org) : `npm install -g @vue/cli`
- Avec NuxtJS (nuxtjs.org) : comprend Vue Router, Babel, SSR...
- Avec le blueprint JHipster : `npm install -g generator-jhipster-vuejs` puis `jhipster --blueprint vuejs`

ARCHITECTURE MVVM

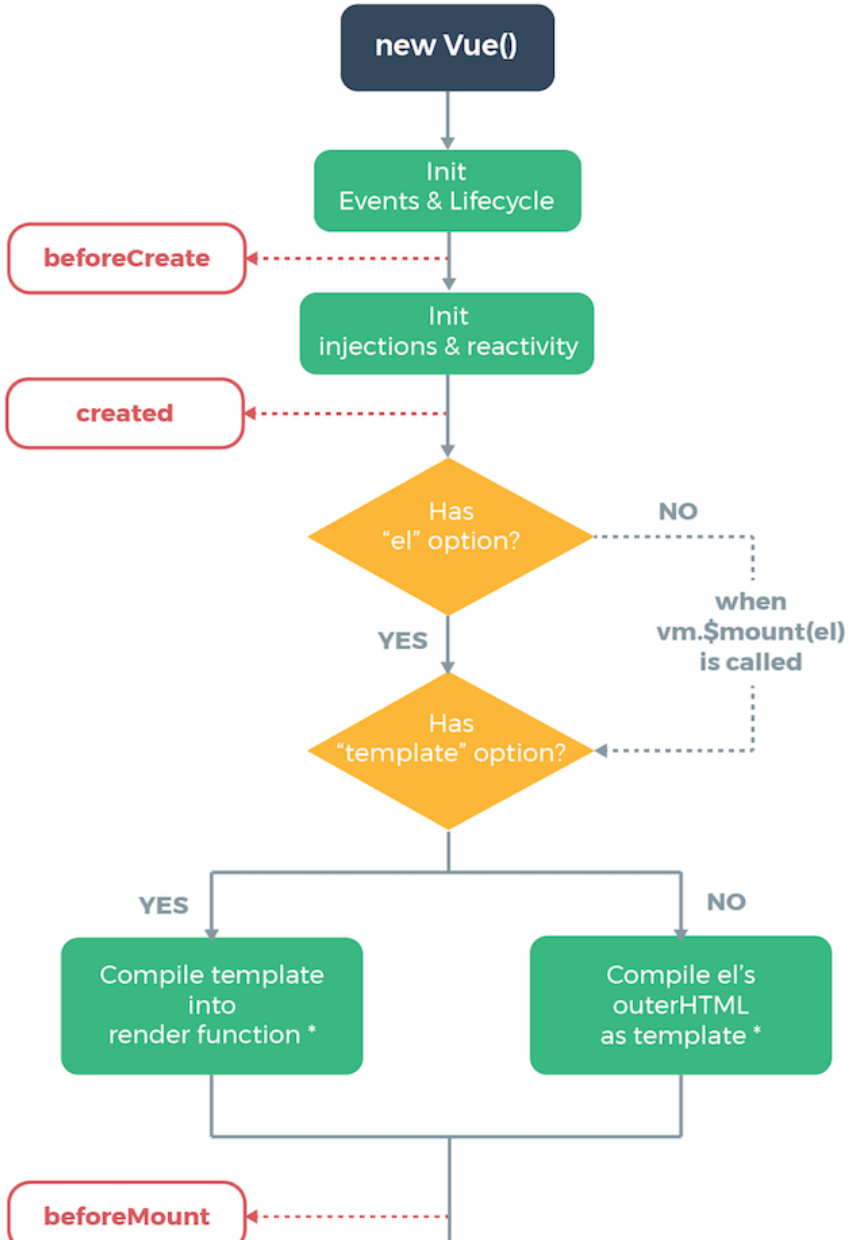


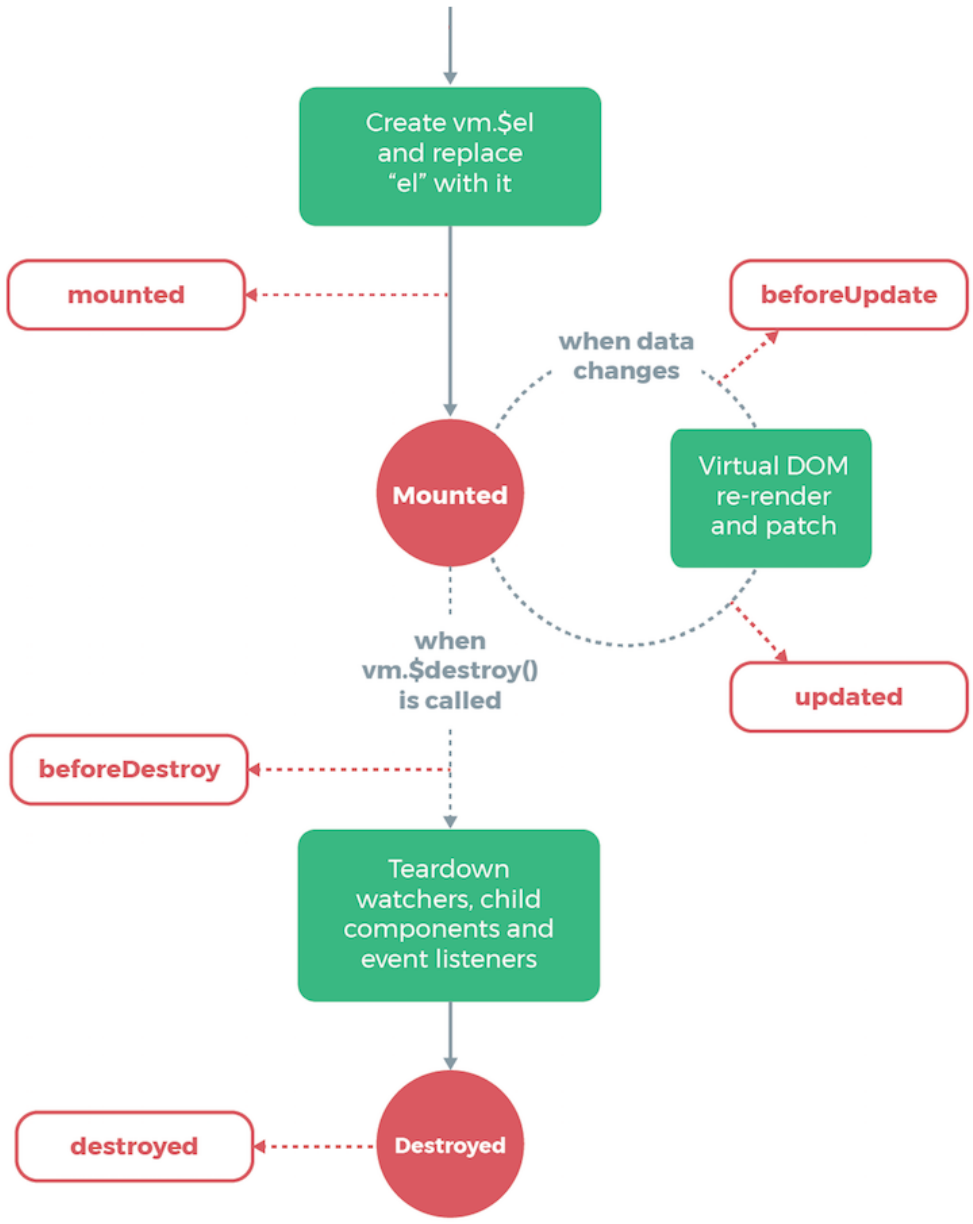
COMPOSANTS

- Composant racine = s'instancie avec `new Vue({ el: '#app' })` souvent dans un `main.js` ou un `main.ts`

- Chaque composant est une instance de Vue et donc un ViewModel qui synchronise Model (objets JS) et Vue (DOM)

CYCLE DE VIE


new Vue()
Init
Events & Lifecycle
beforeCreate
Init
injections & reactivity
created
Has
“el” option?
NO
when
vm.$mount(el)
is called
YES
Has
“template” option?
YES
NO
Compile template
into
render function *
Compile el’s
outerHTML
as template *
beforeMount


Create vm.$el
and replace
“el” with it
mounted
beforeUpdate
when data
changes
Mounted
Virtual DOM
re-render
and patch
when
vm.$destroy()
is called
updated
beforeDestroy
Teardown
watchers, child
components and
event listeners
destroyed
Destroyed

ATTRIBUTS D'UN COMPOSANT

- Données (réactives) avec `data`
- Méthodes (dont fonctions de rendu avec `render`)
- Propriétés calculées avec `computed` (cache)
- Observateurs avec `watch` (un par champ)
- Propriétés communiquer avec les composants enfants avec `props`

DIRECTIVES

- Proches de AngularJS (`ng`)
- Préfixées avec `v-` : `v-if`, `v-for`, `v-on:click`, `v-bind:*` pour les attributs, `v-html`…
- Exemple dans un template : `<p v-if="seen">Partie visible</p>`
- Abbréviations : `:` pour `v-bind` et `@` pour `v-on`
- Two-way binding : `v-model`

ÉVÈNEMENTS

- Directive `v-on:*` écoute les évènements du DOM, ex : `v-on:click`
- Modificateurs, ex : `<a v-on:click.once="doSomething">`
- Touches du clavier, ex : `<a v-on:keyup.enter="typeSomething">`

DÉCLARATIONS DES COMPOSANTS

- Globalement (avant new Vue) :

```
Vue.component('my-component-name', { /* ... */
})
```

- Localement avec `import` et `export default` (ES2015 - https://babeljs.io/docs/en/learn#modules)

COMPOSANTS MONOFICHIER

BONNES PRATIQUES

- Utiliser les méthodes et computed
- Éviter les composants "fourre-tout"
- Utiliser les slots (déléguer l'affichage au composant parent)
- Tendre vers les renderless components : https://adamwathan.me/renderless-components-in-vuejs/
- Éviter de mettre tout et n'importe quoi dans l'arbre à états de VueX

SERVER-SIDE RENDERING

- Par défaut les composants Vue produisent et manipulent du DOM dans le navigateur côté client
- SSR = possibilité de rendre ces composants côté serveur sous forme de chaînes de caractères HTML

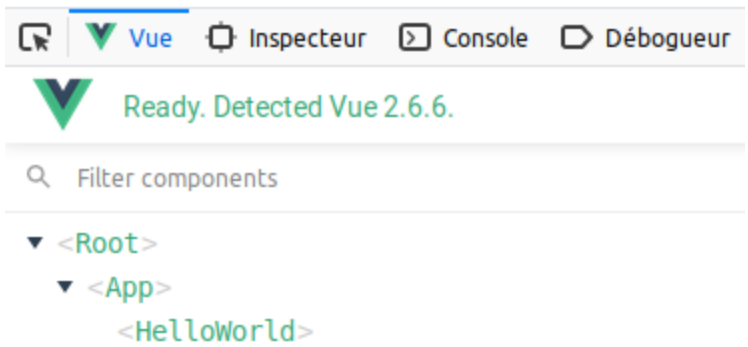
INCONVÉNIENTS SSR

- Nécessite une adaptation pour certaines librairies tierces
- Contrairement à une SPA entièrement statique, besoin de Node.js avec Express par exemple
- Plus de charge côté serveur

AVANTAGES SSR

- Temps de chargement réduit pour les connexions faibles et les vieux navigateurs
- Optimisation SEO par rapport à simple SPA (Single Page Application) = penser prerendering (Webpack + prerender-spa-plugin) ?

VUE DEVTOOLS



- Extension pour les navigateurs
- Voir les composants et leurs données
- https://github.com/vuejs/vue-devtools

Démo !

https://github.com/fdelbrayelle/midi-tech-vuejs/demo

ET LE TURFU ?

- Vue.js 3 au 3ème trimestre 2019
- Contribuez ! https://github.com/vuejs
- Meetups et conférences (events.vuejs.org)

LIENS UTILES

- CodeSandBox : https://codesandbox.io/s/vue
- Vue.js 2 : https://fr.vuejs.org/v2/guide
- NuxtJS : https://nuxtjs.org/guide
- Vue Router : https://router.vuejs.org/guide
- VueX : https://vuex.vuejs.org/guide
- SSR : https://ssr.vuejs.org

Merci !

https://github.com/fdelbrayelle/midi-tech-vuejs