



DÉCOUVRIR VUE.JS PAR LA PRATIQUE

github.com/BudGW/vue-philosophers.git
github.com/BudGW/philosophers-kompanion.git

PRÉSENTATION

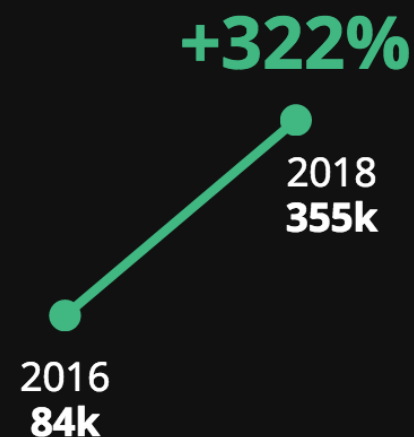
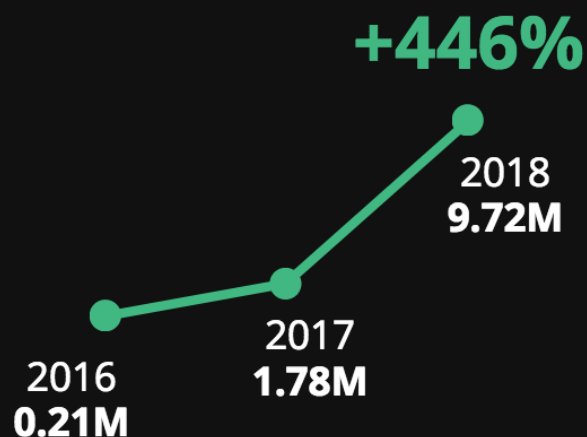
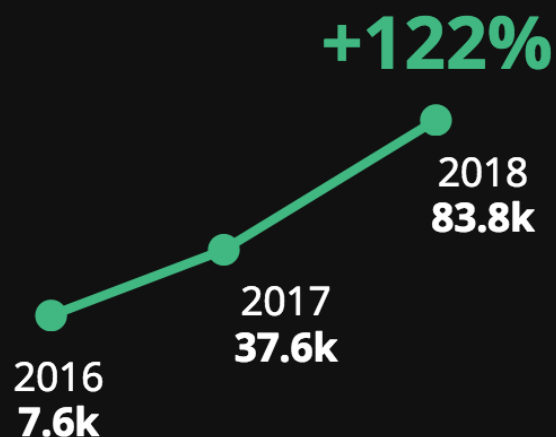
Charles Bouttaz - @CharlesBouttaz - ZenDev
Sylvain Bonnard - @BudGW - OkamiDev



VUE.JS

Progressive Javascript Framework

HISTORIQUE

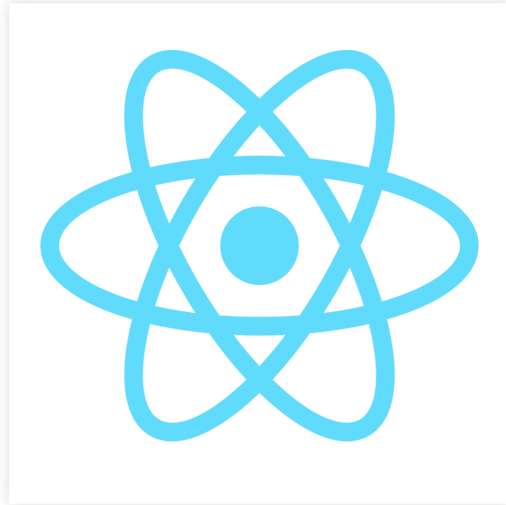


LIBRAIRIE OU FRAMEWORK ?

VIEW LAYER

+

SUPPORT LIBRARIES



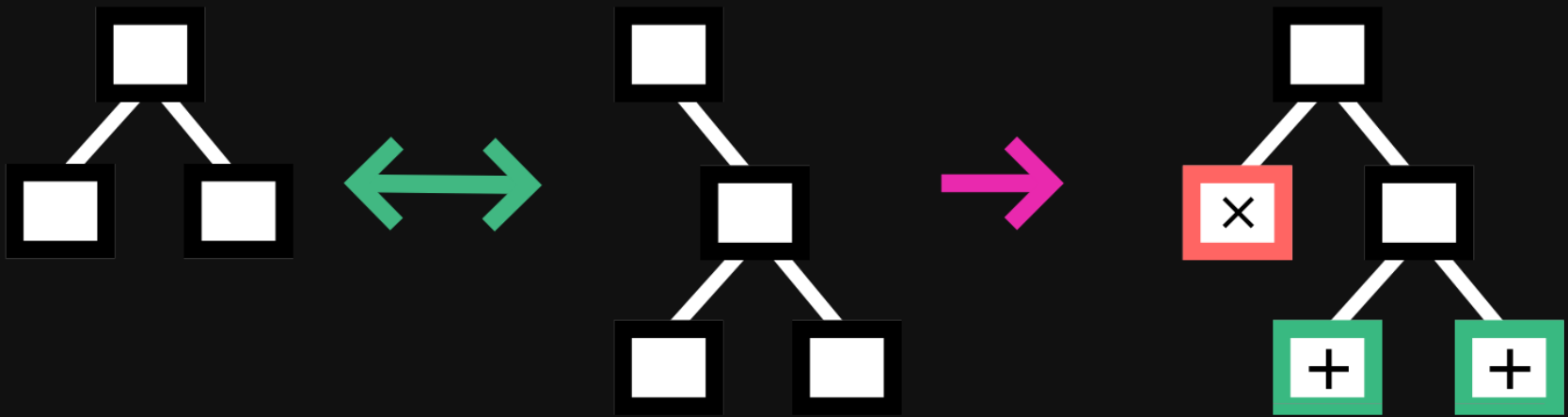
FONCTIONNEMENT GÉNÉRAL

DECLARATIF

Associer un state à un template et le rattacher au DOM

Mise à jour du DOM automatique

Virtual DOM



VIRTUAL

REAL DOM



TEMPLATES

HTML

JSX

Pure javascript

ET PLUS ENCORE...

Watchers

Propriétés calculées

Server side rendering

...

STEP 00

Déclarer un composant

```
Vue.component('logo', {
  template: `<div>{{greeting}} - {{name}}</div>`,
  props: {
    greeting: {
      type: String,
      default: ''
    }
  },
  data: function() { return { name: "John" } }
})
new Vue({
  el: "#app"
})
```

PROPS & DATA

Data : état interne du composant

Props : paramètres d'entrée du composant

```
<logo v-bind:text="myVar"></logo>  
<logo :text="myVar"></logo>
```

ITÉRER

```
<div v-for="item in items">  
  This is {{item.name}}  
</div>
```


STEP 01

[Home](#) | [About](#)

Philosophers list



Maurice Moss



Sheldon Cooper



Jeff Goldblum

SINGLE FILE COMPONENT (SFC)

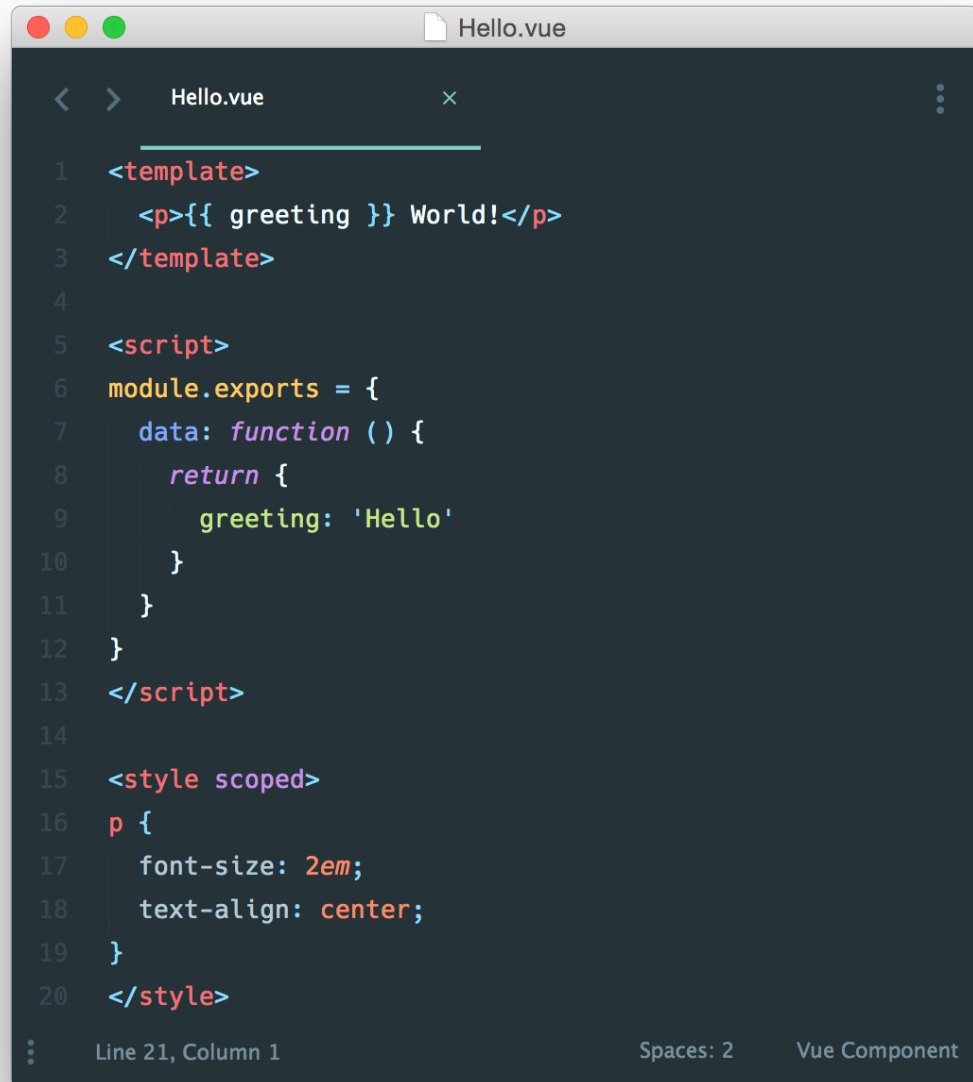
LES PROBLÈMES

Définitions globales des composants

String templates

CSS des composants éloigné de leur définition

Pas de build



```
1 <template>
2   <p>{{ greeting }} World!</p>
3 </template>
4
5 <script>
6   module.exports = {
7     data: function () {
8       return {
9         greeting: 'Hello'
10      }
11    }
12  }
13 </script>
14
15 <style scoped>
16   p {
17     font-size: 2em;
18     text-align: center;
19   }
20 </style>
```

Line 21, Column 1 Spaces: 2 Vue Component

VUE CLI

Bootstrap rapide pour tester...
mais pas que.

```
vue create myproject
```

```
vue serve [filename]
```

```
vue build [filename]
```

```
vue invoke [pluginname]
```

```
vue inspect
```

```
vue init
```

STEP 02

[Home](#) | [About](#) | [Philosophers](#)

Philosophers list



Maurice Moss



Sheldon Cooper



Jeff Goldblum

VUE-ROUTER

DÉFINIR DES ROUTES

```
const Foo = { template: '<div>foo</div>' }  
const Bar = { template: '<div>bar</div>' }  
import Home from './views/Home.vue'  
  
const routes = [  
  { path: '/foo', component: Foo },  
  { path: '/bar/:my-parameter', component: Bar, props: true },  
  { path: '/', name: 'home', component: Home }  
]
```

IMPORTER LE ROUTER

```
const router = new VueRouter({  
  routes  
})  
  
const app = new Vue({  
  router  
}).$mount( '#app' )
```

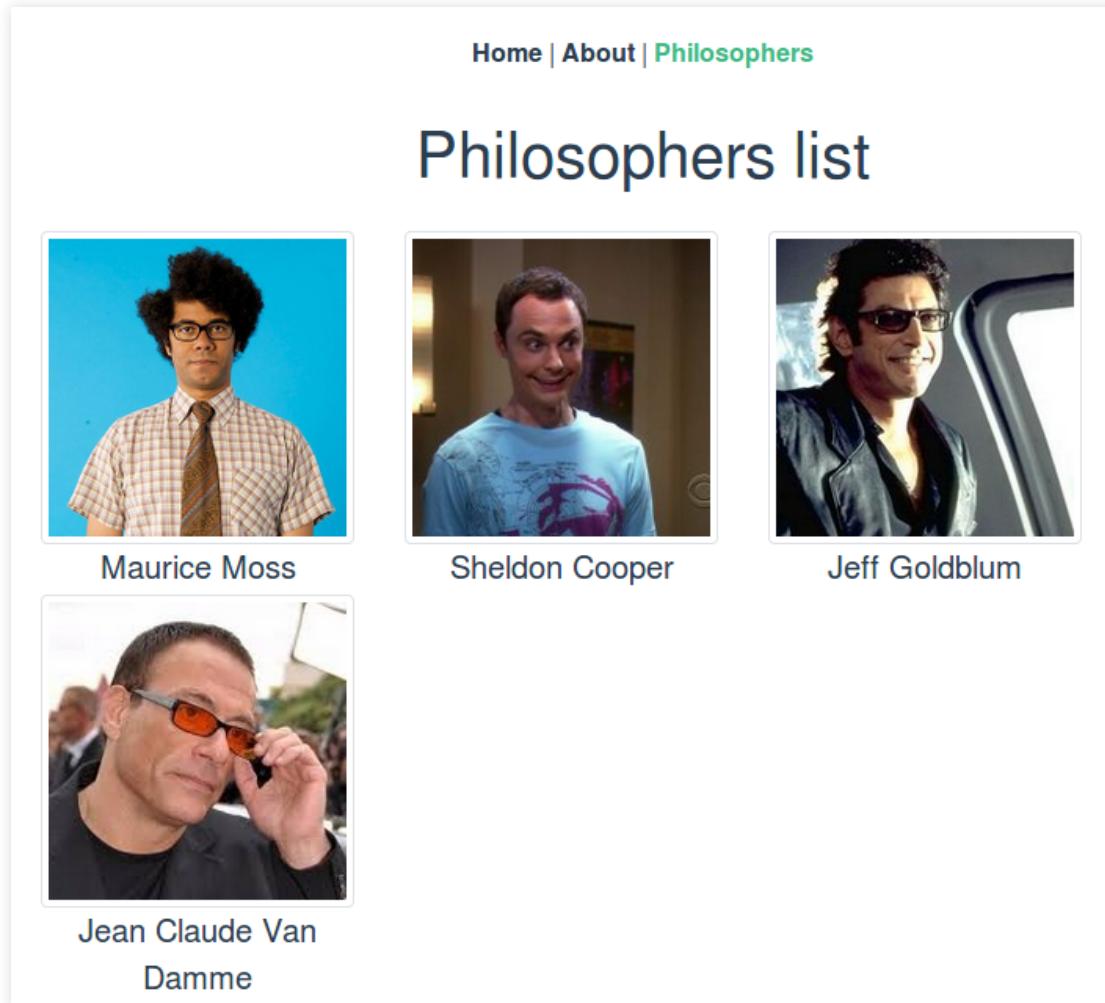
DANS LES TEMPLATES

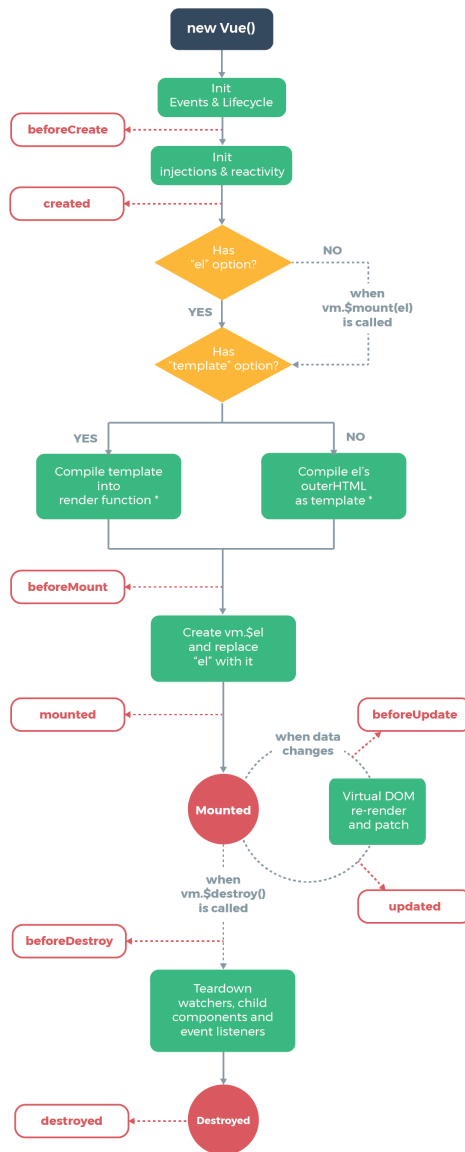
```
<router-link to="/home">Go to Home</router-link>  
<router-link :to="{ name: 'bar', params: { my-parameter: value  
<router-view></router-view>
```

DANS LE CODE

```
router.push('home')  
router.push({ path: 'home' })  
router.push({ name: 'user', params: { userId: 123 } })
```

STEP 03: COMPONENT LIFECYCLE & REST API





* template compilation is performed ahead-of-time if using a build step, e.g. single-file components

COMPUTED PROPERTIES

Vue HTML Result

Edit in JSFiddle



Plus lisibles que du JS intégré au template

Cache

Lazy

STEP 04: COMPONENTS

[Home](#) | [About](#) | [Philosophers](#)

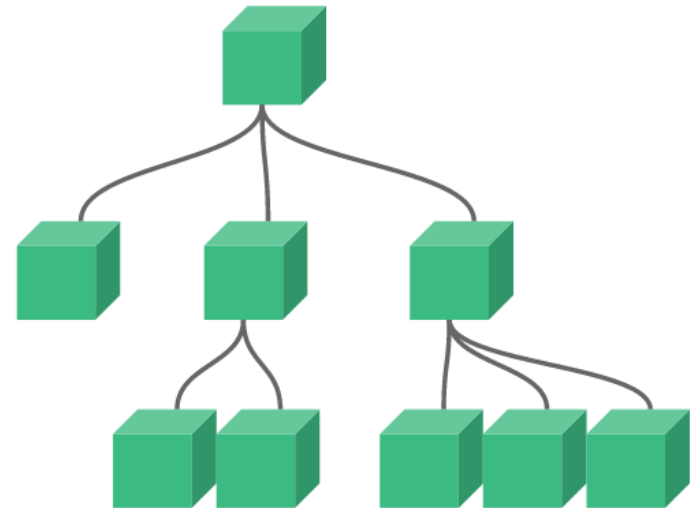
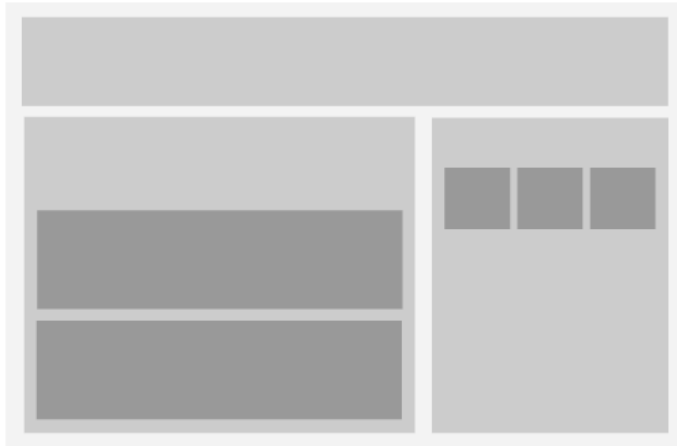


Jean Claude Van
Damme

Moi, Adam et Eve, j'y
crois plus tu vois,
parce que je suis pas
un idiot : la pomme ça
peut pas être
mauvais, c'est plein
de pectine...

Selon les statistiques,
il y a une personne
sur cinq qui est

ORGANISER LES COMPOSANTS EN ARBRES



SLOTS

Paramétrer le contenu d'un composant

Vue HTML Result

Edit in JSFiddle



LAYOUT AVEC DES NAMED SLOTS

```
<!--Déclaration d'un template de composant layout-->
<div class="container">
  <header><slot name="header"></slot></header>
  <main><slot></slot></main>
  <footer><slot name="footer"></slot></footer>
</div>
...
<!--Utilisation du composant dans un autre template-->
<base-layout>
  <template slot="header">
    <h1>Here might be a page title</h1>
  </template>
  <p>A paragraph for the main content.</p>
  <p>And another one.</p>
  <template slot="footer">
```

STEP 05: STYLES & CONDITIONAL RENDERING

[Home](#) | [About](#) | [Philosophers](#)



Maurice Moss

I'm here to drink
milk and kick ass.
And i just finished
my milk...

I'll just put this over
here with the rest of
the fire.

I like being weird,
weird is all I've got.
That and my sweet
style

SCOPED CSS

Evite les conflits de CSS

```
<div class="philosopher-info" data-v-c21348c8="">  
  <p class="disclaimer" data-v-c21348c8="">Fictional character</p>  
</div>
```

```
.philosopher-info[data-v-c21348c8] {  
  width: 800px;  
  margin: 20px auto;  
}
```

PRÉPROCESSEURS CSS

```
<style lang="scss">  
  $couleur-principale: red;  
  
  body {  
    color: $couleur-principale;  
  }  
</style>
```

CONDITIONAL RENDERING


```
<div v-if="type === 'A'">
  A
</div>
<div v-else-if="type === 'B'">
  B
</div>
<div v-else-if="type === 'C'">
  C
</div>
<div v-else="">
  Not A/B/C
</div>

<h1 v-show="ok">Hello!</h1>
```


STEP 06: FORM INPUT BINDING

Home | About | Philosophers

Fictional character



Maurice Moss

Ses citations préférées :

I'm here to drink milk and kick ass. And i just finished my milk...

I'll just put this over here with the rest of the fire.

I like being weird, weird is all I've got. That and my sweet style

A new quote !

Another quote | Add quote

V-MODEL

Binding dans les 2 sens

1 seul par composant

Implémenté par défaut sur les inputs HTML

A implémenter sur ses composants



ECOUTER UN ÉVÈNEMENT

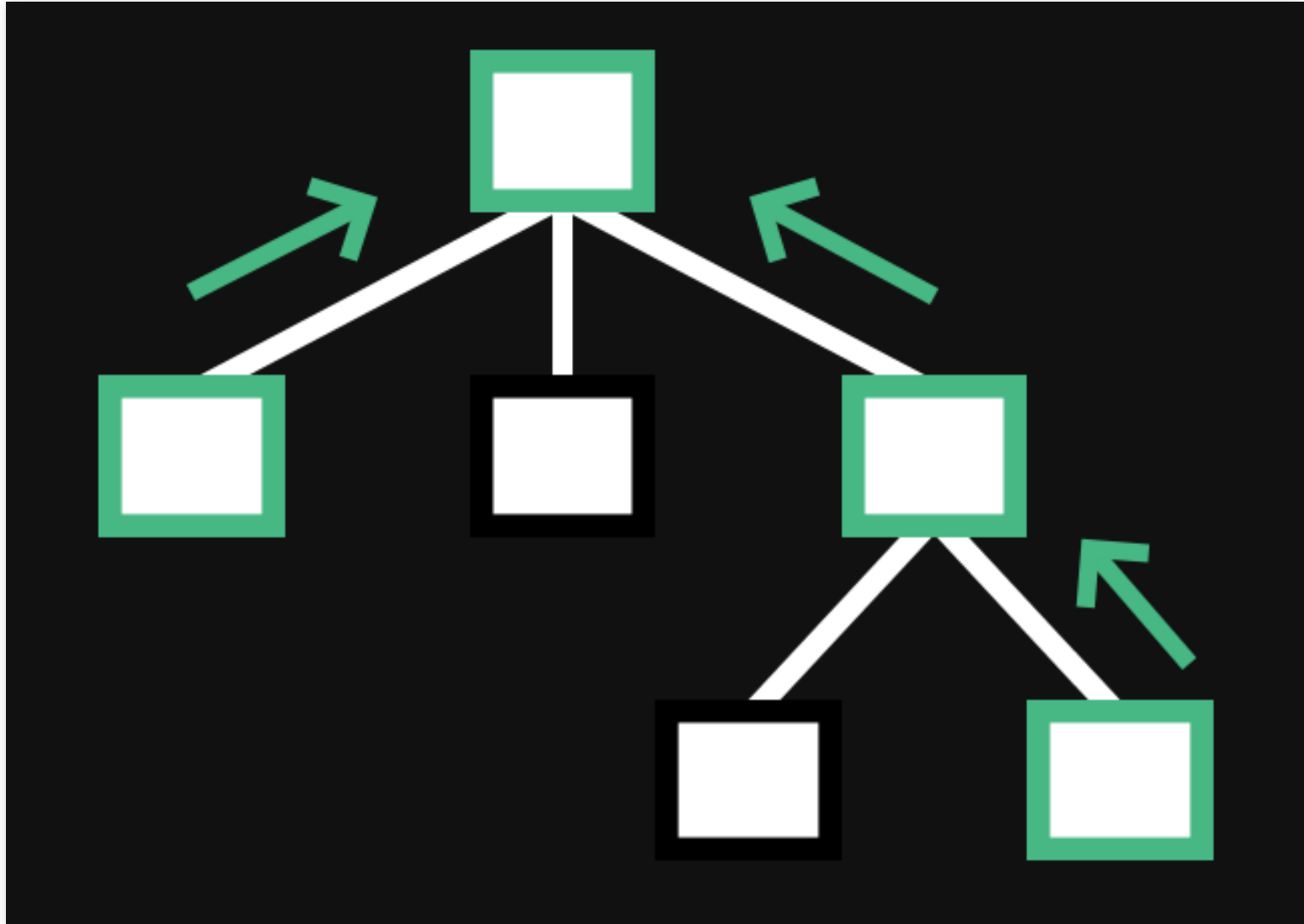
@my-event="onMyEvent"

ou

v-on:my-event="onMyEvent"

STEP 07: COMMUNICATION & EVENTS

EVÉNEMENTS



```
//Fils emmet un évènement
this.$emit('custom-event-name', body)

//Parent écoute un évènement avec v-on ou @ et
//le "brancher" sur une méthode
<my-component v-on:custom-event-name="onCustomEvent">
<my-component @custom-event-name="onCustomEvent">

onCustomEvent(body) {
    ...
}

</my-c
```

STEP 08: CUSTOM INPUTS

IMPLÉMENTER UN V-MODEL

Vue HTML Result

Edit in JSFiddle



STEP 09: TESTING & JEST

JEST

Tests sans configuration (ou presque)

VUE-TEST-UTILS

Mises à jour synchrones

Crée un wrapper avec méthodes utilitaires

Encore en beta (mais plutôt stable)

Instancier le composant

```
const wrapper = shallow(MyComponent)  
//ou  
const wrapper = mount(MyComponent)
```

```
//Modifier une prop  
wrapper.setProps({ myProp: valeur })
```

```
//Modifier une data  
wrapper.setData({ myData: valeur })
```

```
//Accéder à une prop/data  
wrapper.vm.myData
```

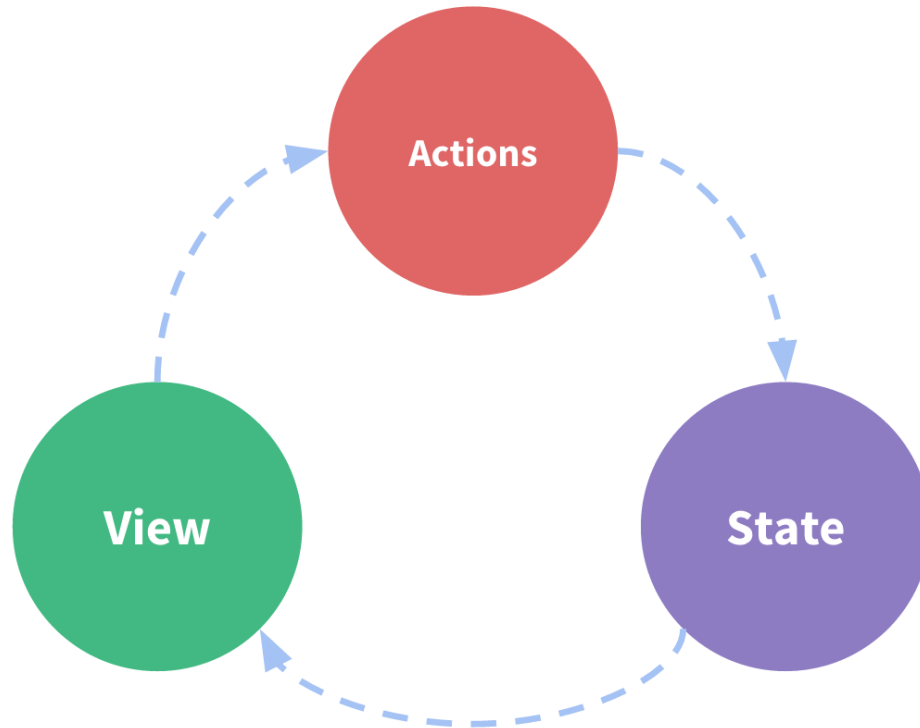
```
//Tester l'évènement input  
expect(wrapper.emitted().input.length).toBe(1)  
expect(wrapper.emitted().input[0]).toEqual([expected])
```

STEP 10: STATE MANAGEMENT WITH VUEX

Comment gérer l'état de nos composants ?

AVEC 1 COMPOSANT

```
new Vue({  
  // state  
  data () {  
    return {  
      count: 0  
    }  
  },  
  // view  
  template: `  
    <div>{{ count }}</div>  
  `,  
  // actions  
  methods: {  
    increment () {  
      this.count++  
    }  
  }  
})
```

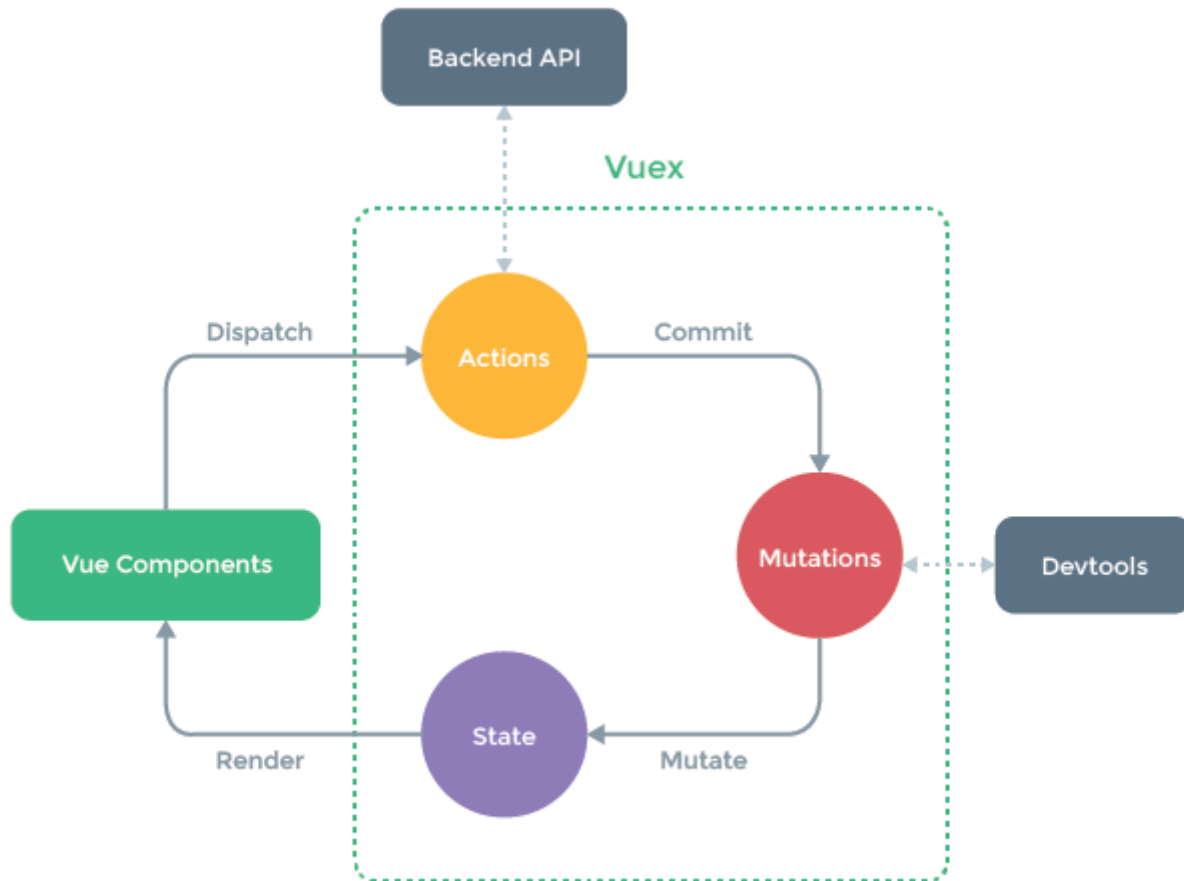


AVEC N COMPOSANTS

Synchroniser via les props & events ?

Synchroniser via un event bus ?

VUEX



WEB COMPONENTS

Custom Elements

Shadow DOM

HTML imports

HTML Template

WEB COMPONENTS

Custom Elements \Leftrightarrow Single File Components

Shadow DOM \Leftrightarrow Virtual DOM

HTML imports \Leftrightarrow JS imports

HTML Template \Leftrightarrow Template SFC

CONCLUSION

LIENS