



**Bienvenue à la Sfeir School
VUE.JS 200**

**WIFI : SFEIRGUEST
MDP : GUEST\$2014**



WIFI : SfeirBeneluxWifi

MDP : xxx



Déroulement de la formation

C'est quand la pause ?
Quand est-ce qu'on mange ?
Tour de table...

Feuille de présence (obligatoire)

Présentation



[sf≡ir]

Ludovic Valente

@ludoo0d0a

Nicolas Frizzarin

@??...

Présentation



[sf=ir]

Cyril Balit

CTO front
@cbalit



Slides de la formation

<http://bit.ly/sfeir-vuejs-200>

(accès restreint)

Déroulement de la formation

Github de la formation

<https://github.com/Sfeir/vuejs-200>

Angular2 People 200

Wassim (GDE)

localhost:4200/#/people

people

Maps List

Found 100 people

Leanne Woodard
BIOSPA

Leanne.Woodard@BIOSPA...
[0784112248](tel:0784112248)

Manager : Erika
Location : SFEIR



Castaneda Salinas
METROZ

Castaneda.Salinas@METR...
[0145652522](tel:0145652522)

Manager : Erika
Location : SFEIR



Phyllis Donovan
PEARLESSA

Phyllis.Donovan@PEARLES...
[0685230125](tel:0685230125)

Manager : Erika
Location : SFEIR



Erika Guzman
CIRCUM

Erika.Guzman@CIRCUM.com
[0678412587](tel:0678412587)

Manager : Mercedes
Location : SFEIR



Moody Prince
TRIPSCH

Moody.Prince@TRIPSCH.c...
[0662589632](tel:0662589632)

Manager : Mercedes
Location : SFEIR



Mercedes Hebert
QUINTITY

Mercedes.Hebert@QUINTIT...
[0125878522](tel:0125878522)

Manager : McLaughlin
Location : SFEIR



Déroulement de la formation

```
$ git clone https://github.com/Sfeir/vuejs-200.git
```

Déroulement de la formation

```
$ (npm install -g @vue/cli)  
$ npm install
```

```
$ npm run client  
$ npm run server
```

npm run client: <http://localhost:8080/>
npm run server: <http://localhost:9000/>

Déroulement de la formation

Un concept clé de Vue.js

Un TP

- **une page d'exercice : stepXX**
<http://localhost:8080/step01>
- **une page de solution : stepXX_solution**
http://localhost:8080/step01_solution

Quickstart

Hello ...

- le fichier source de vuejs
- un template
- un script de démarrage

Les sources

2 types

- standalone:
 - inclut le template compiler
- runtime:
 - utilise uniquement les fonctions render
- CDN, NPM, Bower

Les sources

```
<html>  
  <head>  
    <script src="/path/to/vue.js"></script>  
  </head>  
  <body>  
    <div id="app"></div>  
  </body>  
</html>
```

Le script

- Instancier une Vue avec des options
 - **el** pour cibler l'élément
 - **data** pour exposer des propriétés

```
new Vue({  
  el: '#app',  
  data: {  
    name: "Cyril"  
  }  
})
```

Le template

- Directement dans la page HTML:

```
<html>  
  <head>  
    <script src="/path/to/vue.js"></script>  
  </head>  
  <body>  
    <div id="app">  
      hello {{ name }}  
    </div>  
  </body>  
</html>
```

Le template

- En option à l'instance de vue

```
new Vue({  
  el: '#app',  
  template: '<span>{{name}}</span>',  
  data: {  
    name: "Cyril"  
  }  
})
```

Le template

- en utilisant des script avec type="x-template"

```
<script type="x-template" id="appTpl">  
  <span>Hello {{name}}</span>  
</script>  
  
...  
  
new Vue({  
  el: '#app',  
  template: '#appTpl',  
  ...  
})
```

1

Exercice 1 : prise en main

localhost:8080/step01

Mise en place de votre première application

- Dans le fichier main.js vous allez
 - créer votre première instance Vue
 - elle expose une propriété name
- Dans le fichier index.html
 - utilisez cette propriété directement dans le html

SOLUTION

localhost:8080/step01_solution

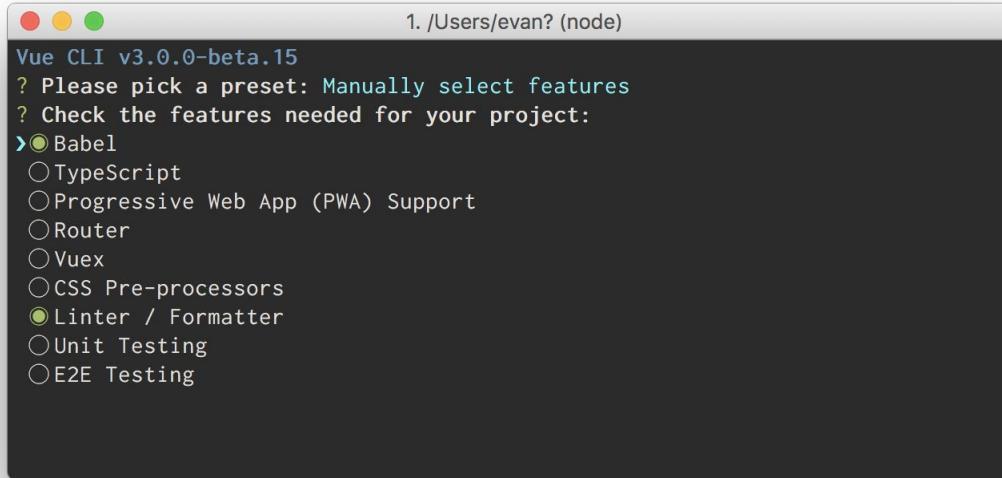
Un peu de tooling

La stack “officielle”

Vue CLI

<https://cli.vuejs.org/>

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



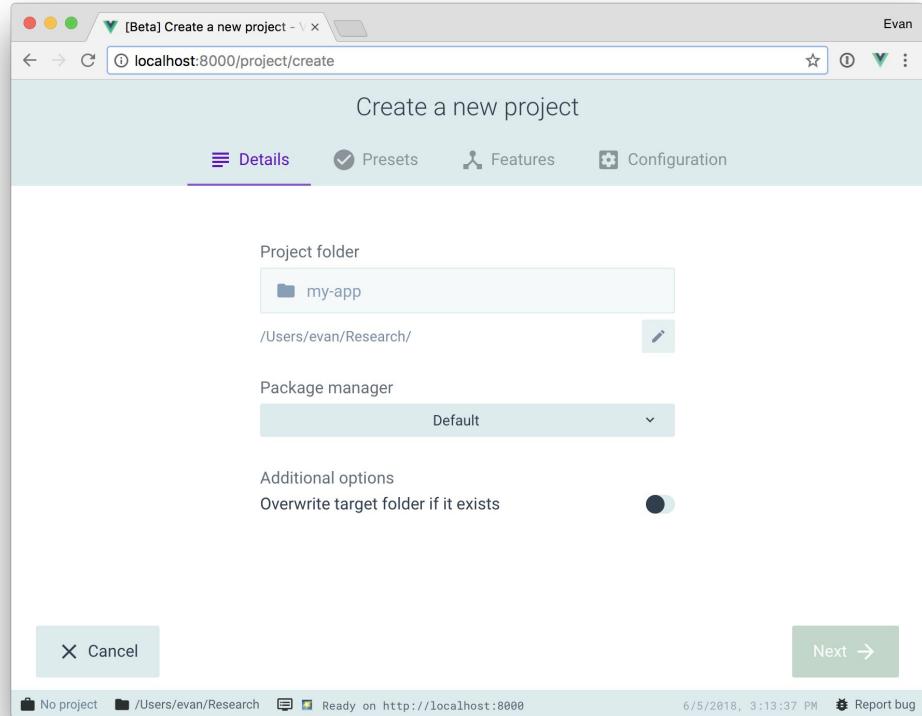
A screenshot of a terminal window titled "1. /Users/evan? (node)". The window displays the following text:

```
Vue CLI v3.0.0-beta.15
? Please pick a preset: Manually select features
? Check the features needed for your project:
>● Babel
○ TypeScript
○ Progressive Web App (PWA) Support
○ Router
○ Vuex
○ CSS Pre-processors
● Linter / Formatter
○ Unit Testing
○ E2E Testing
```

The "Babel" and "Linter / Formatter" options are selected, indicated by a green circle with a dot.

La stack “officielle”

- vue
- vue-cli-service
- vue ui



La stack “officielle”

```
$ vue create <project-name>
```

- plugins
- preset

```
Usage: create [options] <app-name>

create a new project powered by vue-cli-service

Options:

-p, --preset <presetName>      Skip prompts and use saved or remote preset
-d, --default                     Skip prompts and use default preset
-i, --inlinePreset <json>         Skip prompts and use inline JSON string as preset
-m, --packageManager <command>  Use specified npm client when installing dependencies
-r, --registry <url>             Use specified npm registry when installing dependencies
-g, --git [message|false]          Force / skip git initialization, optionally specify message
-f, --force                        Overwrite target directory if it exists
-c, --clone                        Use git clone when fetching remote preset
-x, --proxy                        Use specified proxy when creating project
-h, --help                         output usage information
```

La stack “officielle”

\$ vue-cli-service

- serve
- build
- inspect

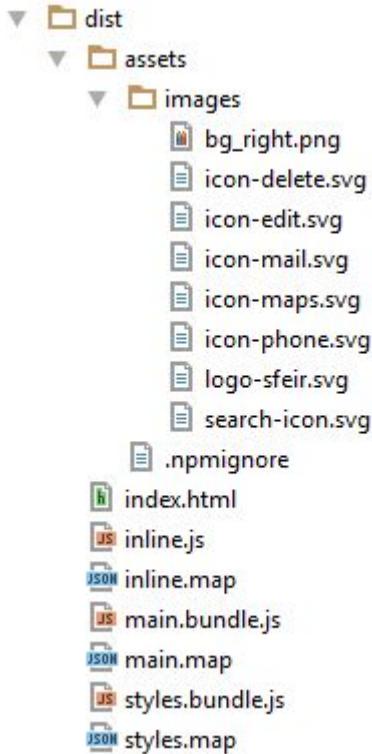
```
Usage: vue-cli-service build [options] [entry|pattern]

Options:

  --mode      specify env mode (default: production)
  --dest      specify output directory (default: dist)
  --modern    build app targeting modern browsers with auto fallback
  --target    app | lib | wc | wc-async (default: app)
  --name      name for lib or web-component mode (default: "name" in package.json or
  --no-clean   do not remove the dist directory before building the project
  --report     generate report.html to help analyze bundle content
  --report-json generate report.json to help analyze bundle content
  --watch      watch for changes
```

WebPack*

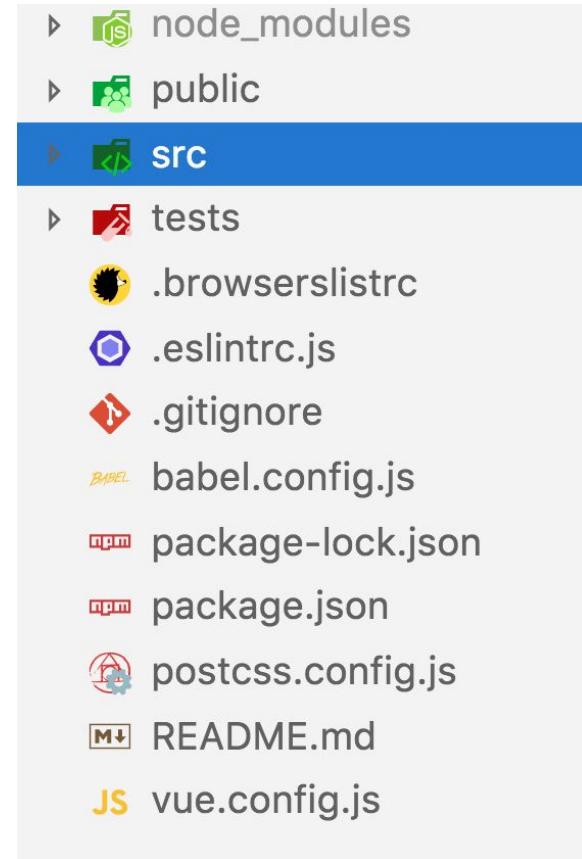
- Bundle en javascript
- hot reload



Configuration

- Les commandes
 - npm run serve
 - npm run build
 - npm run test:unit
 - npm run e2e
- Configuration
 - dev, prod...
 - variables d'environnements dans des fichiers .env

<https://cli.vuejs.org/guide/mode-and-env.html#modes>



Un mot sur le Material Design



Introduction

Getting Started

Themes

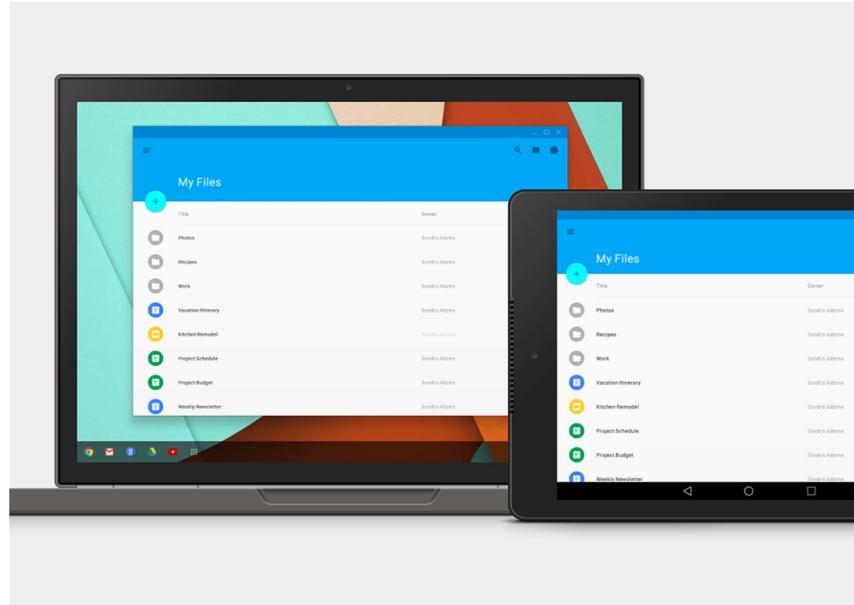
Components

UI Elements

Changelog

About

Introduction



Build well-crafted apps with Material Design and Vue 2

GETTING STARTED

GITHUB

Un mot sur le Material Design

- Le TP utilise des composants Material Design (vue-material)
 - **md-dialog**
 - **md-button**
 - **card-panel**
 - **md-input**
 - ...
- Pour l'utiliser
 - Vue.use(VueMaterial)
- Vous n'êtes pas obligé d'utiliser ces composants

Intégration simple vs SPA

- A tout moment vous pouvez utiliser votre instance (et vos composants)
 - directement dans votre application
 - inclusion des sources javascript dans votre page html
 - en mode spa
 - toute l'application est en vuejs
 - on utilise les outils (webpack, cli)
 - On peut alors utiliser les imports et les exports

2

Exercice 2 : Les import



localhost:8080/step02

Utilisons les imports

- Même exercice mais en utilisant les imports
 - importer Vue
 - VueMaterial
- N'oubliez pas d'indiquer à Vue que vous allez utiliser le Material
 - Vue.use(VueMaterial)
- Nous allons afficher notre message dans un composant **md-card**

```
<md-card>  
  <md-card-header>  
    <div class="md-title">Hello {{name}}</div>  
  </md-card-header>  
</md-card>
```

SOLUTION

localhost:8080/step02_solution

Instance Vue

L'instance:

- Un constructeur
 - des options
 - le sélecteur
 - propriétés
 - méthodes
 - computed
 - lifecycle
 - ...
- Une API
- on peut l'étendre

```
var vm = new Vue({  
  // options  
})
```

```
var MyComponent =  
Vue.extend({  
  ...  
})  
var myComponentInstance =  
new MyComponent()
```

Les propriétés

- exposées dans la map **data**
- réactives
 - Si elles sont définies à l'instanciation
- Propriétés d'instances
 - \$el
 - \$data

```
var vm = new Vue({  
  el: '#example',  
  data: { a: 1 }  
})
```

```
vm.$data
```

Les computed properties

- exposées dans la map **computed**
- pour masquer la logique
- fait référence à des propriétés
- peut être utilisée dans le template
- mise à jour quand les propriétés changent
- getter et setter

```
var vm = new Vue({  
  data: {  
    firstName: 'John',  
    lastName: 'Doe'  
  },  
  computed: {  
    fullName: function () {  
      return this.firstName + ' ' +  
        this.lastName  
    }  
  }  
})
```

Lifecycle

- Méthodes du cycle de vie
 - created
 - mounted
 - beforeUpdate
 - ...
- A implémenter directement dans les options

```
var vm = new Vue({  
  data: {  
    a: 1  
  },  
  created: function () {  
    console.log('a is: ' + this.a)  
  }  
})
```

Les méthodes

- exposées dans la map **methods**
- peut être utilisée dans le template
- méthodes d'instances
 - render
 - prend createElement en paramètre
 - construit un VNode
 - \$on
 - \$emit
 - ...

<https://vuejs.org/v2/api/#Instance-Methods-Events>

```
var vm = new Vue({  
  methods: {  
    doSomething: function() {  
      console.log(this.name);  
    }  
  },  
  render: function (createElement) {  
    return createElement('h' + this.level)  
  },  
})  
  
vm.$on(...)
```

Les watchers

- sur l'instance ou dans une méthodes
- Attention aux arrow function
- opération asynchrones ou complexes
- Attention, privilégier un computed properties

```
var vm = new Vue({  
  created: function () {  
    this.$watch('a', (newVal, oldVal)=> {  
      ...  
    })  
  }  
})  
  
vm.$watch('a', function (newVal, oldVal) {  
  ...  
})
```

computed vs watcher vs method

- une propriété computed est recalculée que si ses dépendances changent
- Une méthode est appelée à chaque fois
- Un watcher permet des traitements plus complexes (asynchrones)

Composants

Composants : arbre

- Arbre de composant
- Les “enfants” sont ajoutés au parent s’ils apparaissent dans son template
- Ils doivent aussi avoir été déclarés dans la section **components**

DANS LE PARENT

```
import MyChild from 'path/to/myChild'

export default {
  template: '<my-child></my-child>',
  components: {
    'my-child': MyChild
  }
}
```

Créer un composant

- Un composant peut être enregistré de façon globale

`Vue.component(tagName, options)`

- des options (identique que pour l'instance)
 - le sélecteur
 - propriétés
 - méthodes
 - computed
 - lifecycle
 - ...
- MAIS data doit retourner une fonction

```
data: function() {  
    return { a: 1 };  
}
```

Créer un composant .vue

- Pourquoi
 - sinon global donc nom unique
 - string template
 - css global, pas de scope
 - pas de build
- dans un même fichier
 - template, script et css scopée
- plugin webpack
 - fournit par le cli



```
<template lang="jade">
  div
    p {{ greeting }} World!
    other-component
  </template>

<script>
  import OtherComponent from './OtherComponent.vue'

  export default {
    data () {
      return {
        greeting: 'Hello'
      }
    },
    components: {
      OtherComponent
    }
  }
</script>

<style lang="stylus" scoped>
  p
    font-size 2em
    text-align center
</style>
```

Line 27, Column 1 Spaces: 2 Vue Component

3

Exercice 3 : Un vrai composant



localhost:8080/step03

Mise en place de votre premier composant vue

- Créer un composant **App** possédant
 - un template
 - une section script
 - une section style
- Importez dans le fichier main.js
 - ce composant
 - Le constructeur Vue
- Vous allez créer une instance de Vue en utilisant la méthode render sur le composant

SOLUTION

localhost:8080/step03_solution

Databinding & template

JavaScript

```
<html>
    Bonjour <span id="name"></span>
    <input type="text"/>
</html>
```

```
window.onload = function(){
    var span = document.querySelector('name');
    var input = document.getElementsByTagName('input')[0];

    input.onkeyup = function(){
        if (span.textContent || span.textContent === "") {
            span.textContent = input.value;
        } else if(span.innerText || span.innerText === "") { // IE
            span.innerText = input.value;
        }
    };
};
```

jQuery

```
<html>
    Bonjour <span id="name"></span>
    <input type="text"/>
</html>

$(document).ready(function() {
    var $input = $('input');
    var $span = $('#name');

    $input.keyup(function (event) {
        $span.text(event.target.value);
    });
});
```

Vuejs

```
<div>  
  <input type="text" name="myName" v-model="myName">  
  <p>Bonjour {{myName}}</p>  
</div>
```

Syntaxe

Properties, events et interpolation

```
<div> My name is {{name}} </div>
```

```
<button v-on:click="changeName" v-bind:disabled="isDisable">
```

```
</button>
```

Interpolation et expression

- Interpolation

```
<div>Hello {{name}}</div>
```

- Les expressions
 - dans le contexte du composant
 - du JS mais
 - une seule expression

Property binding

Type	cible	exemple
propriété	Attribut d'élément Attribut de component	<pre> <my-component v-bind:data="currentData"></my-component></pre>

- raccourci: :attr

```
<div :id="user.id"></div>
```

Event binding

Type	cible	exemple
Évènement	Évènement d'élément Événement de composant	<pre><button v-on:click="onSave"></button> <hero-detail v-on:deleted="onDeleted(arg1,arg2,\$event)"></pre>

- raccourci: **@event**

```
<div @click="do()"></div>
```

- Référence à l'évent grâce à **\$event**

4

Exercice 4 : Afficher une personne

localhost:8080/step04

- Créer un composant **Home.vue** pour afficher les détails d'une personne
- Utiliser les fichiers suivants comme exemple pour le contenu :
 - `src/_static/home.html`
 - `src/_static/home.css`
 - `src/_static/people.js`
- Utiliser les données de `people.js` dans le composant (l'importer)
- Afficher ce nouveau composant dans le composant App

4

Exercice 4 : Afficher une personne

The screenshot shows a web browser window titled "Angular2 People 200". The address bar displays "localhost:4200/#/home". The page has a blue header with the word "people" on the left and "Maps" and "List" buttons on the right. Below the header, there is a card for a person named "Vargas Shaffer" from "Sfeir-Luxembourg". The card includes an email icon and the email address "vargasshaffer@sfeir.com", a phone icon and the phone number "+33197542822", and a location section stating "Manager : Bruno" and "Location : SFEIR". To the right of the card is a circular profile picture of a man. At the bottom right of the card are three icons: a person with a location pin, a pen, and a trash can.

SOLUTION

localhost:8080/step04_solution

Gestion des événements DOM

Les événements

- Nom de l'événement précédé par **v-on**:
- Fait référence à une fonction définie dans **methods**
- Pour récupérer les détails de l'event : **\$event**

```
{  
  methods:{  
    onClick($event){  
      this.values = $event.target.value;  
    }  
  }  
}
```



<input
 v-on:click="onClick">

5

Exercice 5 : gérer un clic

localhost:8080/step05

- Un bouton “random” a été ajouté dans le template du composant Home
- Ajouter un clic sur ce bouton pour afficher une personne au hasard
 - exploiter le tableau PEOPLE fourni

SOLUTION

localhost:8080/step05_solution

Communication Serveur

Usages

On utilise AXIOS. Les méthodes disponibles:

axios.get(url, options);

axios.post(url, data, options);

axios.put(url, data, options);

axios.delete(url, options);

<https://github.com/mzabriskie/axios>

Gérer les retours

```
axios.get(url, options)
  .then((response) => {
    /* do something with response.data*/
  })
  .catch((error)=> {
    console.log(error);
  });
});
```

Exercice 6 : fini les données en dur

6

localhost:8080/step06

Récupérer les personnes à partir du serveur

- Un squelette de service a été ajouté dans le fichier service/PeopleService.js
- Implémenter les méthodes fetch() et fetchRandom()

```
fetch() {  
    return axios.get(BACKEND_URL)  
        .then(response => response.data)  
}
```

- GET <http://localhost:9000/api/people> retourne une collection de PEOPLE
- GET <http://localhost:9000/api/people/random> retourne une personne au hasard
- Importer ce service dans le composant Home pour l'utiliser

SOLUTION

localhost:8080/step06_solution

Navigation

Angular2 People 200 Wassim (GDE)

localhost:4200/#/home

pe@ople

Maps List

Do you know?

Mercedes Hebert
QUINTITY

Mercedes.Hebert@QUINTITY...
 0125878522

Manager : McLaughlin
Location : SFEIR



Angular2 People 200 Wassim (GDE)

localhost:4200/#/people/5763cd4d979b62a209809160

pe@ople

Maps List

Fiche personnelle

Mercedes Hebert
QUINTITY

 [Mercedes.Hebert@QUINTITY...](mailto:Mercedes.Hebert@QUINTITY.COM)

 [0125878522](tel:0125878522)

Manager : [McLaughlin](#)
Location : [SFEIR](#)

Skills

ex commodo pariatur sit aute

Angular2 People 200 Wassim (GDE)

localhost:4200/#/people

Maps List

pe@ople

Found 10 people

Leanne Woodard
BIOSPAN

 [Email](#) [Leanne.Woodard@BIOSPA...](#) [Phone](#) [0784112248](#)

Manager : Erika
Location : SFEIR

[Edit](#) [Delete](#)

Castaneda Salinas
METROZ

 [Email](#) [Castaneda.Salinas@METR...](#) [Phone](#) [0145652522](#)

Manager : Erika
Location : SFEIR

[Edit](#) [Delete](#)

Phyllis Donovan
PEARLESSA

 [Email](#) [Phyllis.Donovan@PEARLES...](#) [Phone](#) [0685230125](#)

Erika Guzman
CIRCUM

 [Email](#) [Erika.Guzman@CIRCUM.com](#) [Phone](#) [0678412587](#)

[+](#)

72 HOOLE

Angular2 People 200 Wassim (GDE)

localhost:4200/#/edit/5763cd4d979b62a209809160

Maps List

people

Update Mercedes Hebert

ID (disabled)
5763cd4d979b62a209809160

First name *
Mercedes

Last Name *
Hebert

Email *
Mercedes.Hebert@QUINTITY.com

Address
Laurel Avenue

City Postal Code
Northchase **85752**

Phone * (ex: 0612345678)
0125878522

Manager

Cancel Save



The screenshot shows a web application for managing people records. The URL in the browser is <http://localhost:4200/#/edit/5763cd4d979b62a209809160>. The page title is "Angular2 People 200" and the user is "Wassim (GDE)". The navigation bar includes links for "Maps" and "List". On the left, there's a sidebar with the word "people". The main content area is titled "Update Mercedes Hebert" and shows a form for editing the record. The form fields are: First name * (Mercedes), Last Name * (Hebert), Email * (Mercedes.Hebert@QUINTITY.com), Address (Laurel Avenue), City (Northchase), and Postal Code (85752). Below the address fields, there's a note about phone numbers. At the bottom of the form is a checkbox labeled "Manager" which is checked. At the very bottom of the page are "Cancel" and "Save" buttons.

Angular2 People 200 Wassim (GDE)

localhost:4200/#/locator

people

Maps List

Map Satellite

Mercedes Hebert

Map data ©2016 Google Terms of Use | Report a map error

SCHOOL 74

Labels on the map include: Houilles, Bezons, Colombes, Gennevilliers, Stade de France, Asnières-sur-Seine, Saint-Ouen, Aubervilliers, Drancy, Livry-Gargan, Clichy, Le Raincy, Villemomble, Gagny, Romainville, Les Lilas, Bagnolet, Montreuil, Rosny-sous-Bois, Fontenay-sous-Bois, Nogent-sur-Marne, Bry-sur-Marne, Boulogne-Billancourt, Saint-Cloud, Tour Eiffel, Bois de Boulogne, Grenelle, Bd Saint-Germain, Bd Sébastien, Bd Arago, Bd Peripherique, St-Maurice, Joinville-le-Pont, and Montrouge.

Configuration (simple)

- **path** : l'URL de route, peut contenir des zones dynamiques (ex: /people/:id)
- **name**: le nom de la route
- **component** : le composant associé à cette route (ex: PeopleComponent)
- **redirect**: le path ou le name de la route vers laquelle rediriger
- **alias**: redirection **sans** changement d'URL

Configuration (avancée)

- **components map** avec le nom de l'emplacement dans lequel le composant doit s'afficher
- **meta** : données associées à la route
- **children** : un tableau de définition des sous-routes

```
routes: [
  {
    path: '/',
    components: {
      default: Foo,
      a: Bar,
      b: Baz
    },
    meta: {
      requiresAuth: true
    }
}]
```

Guards

- Intervenir **avant ou après** un changement de route
- Global sur le router (toutes les routes)
 - beforeEach, afterEach
- Au niveau d'une route
- Au niveau du composant
- Prend en argument
 - to: route cible
 - from: route précédente
 - next: function à invoquer en lui passant
 - rien (ok)
 - un path= redirection
 - false: arrêt
 - error

```
router.beforeEach((to, from, next) => {
  // ...
})

routes: [
  {
    path: '/foo',
    component: Foo,
    beforeEnter:(to, from, next) => {
      // ...
    }
  }
]

const Foo = {
  template: `...`,
  beforeRouteEnter (to, from, next) {
  }
}
```

Configuration simple

```
import VueRouter from 'vue-router'

import Home from './Home.vue'
import Update from './Update.vue'

const routes = [
  {name: 'home', path: '/home', component: Home},
  {name: 'edit', path: '/edit/:id', component: Update},
  {path: '/', redirect: '/home'}
]
export default new VueRouter({routes})
```

Définition au niveau de l'instance

```
import VueRouter from 'vue-router'  
...  
import router from './views/router.js'  
Vue.use(VueRouter)  
  
new Vue({  
  el: '#app',  
  router,  
  render: h => h(App)  
})
```

Stratégie de navigations

- Par “Hash”: Par défaut
 - ex: localhost/#/people/1
- Par “Path”: Mode HTML5 et pushState
 - ex: localhost/people/1
 - attention implique une règle côté serveur

```
const router = new VueRouter({  
  mode: 'history',  
  routes: [...]  
})
```

Utilisation dans un composant : les paramètres

- Pour les routes dynamiques
 - référence \$route.params
 - guard
 - props au composant
 - réutilisable
 - testable

```
{  
  template: '<div>User {{ $route.params.id }}</div>'  
}
```

```
beforeRouteEnter: function(route, from, next){  
  route.params.id  
}  
}
```

```
{  
  props: ['id'],  
}  
//In route configuration  
{  
  { path: '/path/:id', ... , props: true }  
}
```

Utilisation dans un composant : navigation

- récupération du router configuré
- **push**
 - `router.push('home')`
 - `router.push({ path: 'home' })`
 - `router.push({ name: 'user', params: { userId: 123 } })`
 - `router.push({ path: 'register', query: { plan: 'private' } })`
- **replace**: remplacer l'url courante
- **go(n)**: avancer ou reculer dans l'historique

```
import router from './router.js'

export default {
  ...
  methods: {
    goBack: function () {
      router.go(-1);
    }
  }
}
```

Utilisation dans un composant : HTML

```
<router-link :to="{ name: 'edit', params: { id: person.id }}">  
    <md-icon class="md-accent">mode_edit</md-icon>  
</router-link>  
  
<section class="container">  
    <router-view></router-view>  
</section>
```

7

Exercice 7 : une navigation côté client

localhost:8080/step07

- Compléter le fichier `src/views/router.js` avec la configuration des routes
- Mettre à jour le fichier `src/main.js`
 - Importer et utiliser le routeur
- Mettre à jour le template du composant App
 - Utiliser le `<router-view></router-view>` dans le fichier `App.vue`

SOLUTION

localhost:8080/step07_solution

Ajoutons des fonctionnalités

Itérer sur une collection avec v-for

- Peut itérer sur:
 - un tableau, un objet, un nombre, une chaîne
- peut utiliser l'opérateur **in** ou **of**
- génère un template par élément
- ***index, key, value***

```
<ul>
  <li v-for="(fruit,index) in fruits">
    {{ index }} : {{ fruit.name }}
  </li>
</ul>
```

```
<ul>
  <li v-for="(key,value) of myobject">
    {{ key }} : {{ value }}
  </li>
</ul>
```

8

Exercice 8 : Répéter les personnes

localhost:8080/step08

Afficher la liste des personnes en utilisant la directive v-for

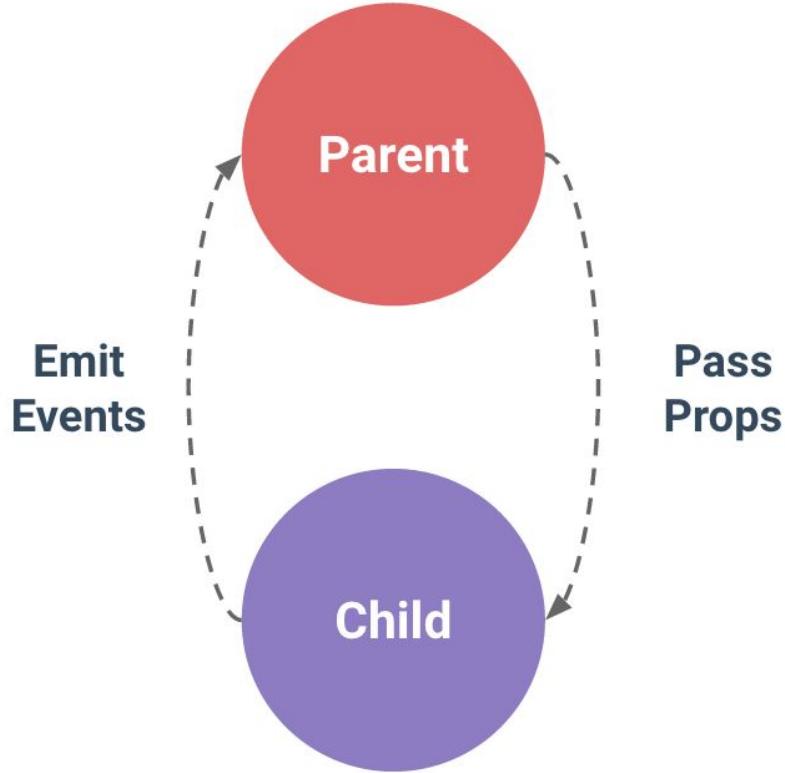
- Créer un composant **People** pour afficher la liste des personnes
- Lui associer une route #/people accessible depuis le lien (en haut dans la toolbar)
- Pour la liste vous pouvez répéter le contenu du composant Home:
 - HTML+CSS
 - note: il est inutile de copier le bouton “random”
- Nous verrons comment améliorer cela lors de la prochaine question

SOLUTION

localhost:8080/step08_solution

Communication entre composants

Composant : event and props



Les props

- Passées dans la propriété **props**
- camelCase vs kebab-case
- Tableau de chaînes
 - nom de la propriété

```
export default {  
  props: ['message'],  
  template: '<span>{{ message }}</span>'  
})
```

```
<my-child message="Hello"></my-child>  
<my-child v-bind:message="myMessage"></my-child>
```

Les props avec contraintes

- Map

- nom de la prop
- contraintes
 - type
 - required
 - validateur
 - custom

```
props: {  
  propA: Number,  
  propB: [String, Number],  
  propC: {  
    type: String,  
    required: true  
  },  
  propD: {  
    type: Number,  
    default: 100  
  },  
}
```

```
props: {  
  propE: {  
    type: Object,  
    default: function () {  
      return { message: 'hello' }  
    }  
  },  
  propF: {  
    validator: function (value) {  
      return value > 10  
    }  
  }  
}
```

Les events

- Le composant enfant dispatch avec `this.$emit`
- Le composant parent écoute avec `v-on` ou `@`

IN CHILD

```
methods: {  
  doSomething: function () {  
    this.$emit('custom', this.data)  
  }  
}
```

IN PARENT

```
template: '<my-child  
v-on:custom="onCustom"></my-child>'  
methods: {  
  onCustom: function (data) {}  
}
```

Exercice 9 : Réutilisation des composants

9

localhost:8080/step09

Créer un composant pur pour éviter la duplication

- Créer un composant **CardPanel** qui affichera les détails d'une personne
 - note: Ce composant sera créé dans le répertoire **components** car il sera utilisé à plusieurs endroits
- Y transférer les contenus HTML et CSS dupliqués
- Faire en sorte que les composants **Home** et **People** l'utilisent comme cela:

```
<sfeir-card :person="person"></sfeir-card>
```

SOLUTION

localhost:8080/step09_solution

Exercice 10 : supprimer une personne

10

localhost:8080/step10

Créer un composant pour éviter la duplication

- Ajouter un événement clic sur le bouton de suppression dans **CardPanel**
- Propager l'événement
 - l'événement devra s'appeler **personDelete**
 - astuce : utiliser **\$emit**
 - dans **People** supprimer l'élément de la liste
 - dans **Home** changer de card (comme le random)
- L'API à utiliser est celle-ci:
 - DELETE <http://localhost:9000/api/peoples/:id>
 - retourne une collection de personnes à jour

SOLUTION

localhost:8080/step10_solution



Si vous appréciez la formation, Envoyez un Tweet !

#sfeirschool #vuejs
@sfeir @cbalit



Si vous appréciez la formation, Envoyez un Tweet !

#sfeirschool #vuejs
@sfeir @SFEIRLux @ludoo0d0a

Vers des concepts plus avancés

Formulaires et validations

HTML



Les formulaires: 2 problématiques

- Collecter les données
- validation des données saisies

Collecter les données

Une directive : v-model

```
<div>  
  <input type="text" name="myName" v-model="myName">  
  <p>Bonjour {{myName}}</p>  
</div>
```

v-model

- Sur input, textarea, checkbox, radio et select
- Peut combiner avec v-for pour construire les options
- Pour les checkbox ou radio, utiliser v-bind pour associer une valeur

```
<input  
  type="checkbox" v-model="toggle"  
  v-bind:true-value="a"  
  v-bind:false-value="b"  
>
```

```
<input type="radio" v-model="pick"  
  v-bind:value="a">
```

11

Exercice 11 : ajouter une personne (1/2)

localhost:8080/step11

- Petite contrainte : affichage dans une modale
- Le composant **People** a été complété avec deux méthodes :
 - **showDialog()** : permet d'afficher la modale
 - **hideDialog()** : permet de cacher la modale
- Le template a été complété avec :
 - un bouton pour ajouter une personne (affiche la modale d'ajout)
 - l'HTML de la modale
 - l'emplacement du formulaire d'ajout... (voir slide suivant)

11

Exercice 11 : ajouter une personne (2/2)

- Créer un composant **Form** (dans components)
 - vous trouverez dans src/_static les fichiers HTML et CSS à utiliser
 - Propagation d'événement custom: **cancel, save**
- Mettre à jour le composant **People** en y intégrant **Form**
- Implémenter dans le **People** la méthode **add** qui ajoute un contact
 - L'API à utiliser est celle-ci:
 - POST <http://localhost:9000/api/people/>
 - retourne la personne créée

SOLUTION

localhost:8080/step11_solution

12

Exercice 12 : modifier une personne (1/2)

localhost:8080/step12

- Créer un composant **Update** (dans views)
- Ce composant doit être accessible via l'url `#/edit/:id`
 - pensez à mettre à jour `src/views/router.js`
 - ainsi que `src/components/CardPanel.vue`
- Récupérer le paramètre `id` depuis la route
- Utiliser l'API `/api/peoples/:id` (GET)
 - pensez à rajouter la méthode `fetchOne` dans le `peopleService`

12

Exercice 12 : modifier une personne (2/2)

- Utiliser **Form** dans **Update** pour afficher le formulaire
 - props: person
- Mettre à jour **Form** en ajoutant un “mode édition”
 - L'idée est d'utiliser le même formulaire pour l'édition et la création
 - Si **person** alors **editMode=TRUE**
- La mise à jour se fait sur l'API /api/people/:id (PUT)
 - pensez à rajouter la méthode **update** dans le peopleService

SOLUTION

localhost:8080/step12_solution

La validation

2 approches

Solutions non supportées par VueJs :

- **Template driven form : vee-validate**
- **Model driven form: vuelidate**

<https://vuelidate.netlify.com>

<https://baianat.github.io/vee-validate/>

Les états d'un contrôle

- **pristine** : l'utilisateur n'a pas interagi avec le contrôle
- **dirty** : l'utilisateur a déjà interagi avec le contrôle
- **valid** : le contrôle est valide
- **invalid** : le contrôle n'est pas valide
- **touched** : le contrôle a perdu le focus
- **untouched** : le contrôle n'a pas encore perdu le focus

Template driven form

- Ajouter la dépendance vee-validate

```
npm install vee-validate --save
```

- Ajouter vee-validate au projet

```
import Vue from 'vue';
import VeeValidate from 'vee-validate';

Vue.use(VeeValidate);
```

Configuration

```
const config = {  
    locale: en,  
    dictionary: {en:'...',fr:'...'},  
    strict: true,  
    classes: false,  
    classNames: {  
        touched: 'touched', // the control has been blurred  
        untouched: 'untouched', // the control hasn't been blurred  
        valid: 'valid', // model is valid  
        invalid: 'invalid', // model is invalid  
        pristine: 'pristine', // control has not been interacted with  
        dirty: 'dirty' // control has been interacted with  
    }  
};  
Vue.use(VeeValidate, config);
```

Utilisation des validateurs

- Dans le template HTML
 - v-validate
 - utilise des rules
 - le champs doit posséder un attribut **name** ou **data-vv-name**

```
<div class="column is-12">  
  <label class="label" for="email">Email</label>  
  <p>  
    <input v-validate="'required|email'" name="email" type="text" placeholder="Email">  
  </p>  
</div>
```

Gestion des erreurs

- Objet **errors**

- `first('field')`
- `collect('field')`
- `has('field')`
- `all()`
- `any()`

```
<input v-validate="'required|email'" :class="{'is-danger': errors.has('email') }" name="email" type="text"
placeholder="Email">

<span v-show="errors.has('email')">{{ errors.first('email') }}</span>
```

Les validateurs

- required
- email
- regex
- min
- max
- ...

<http://vee-validate.logaretm.com/#available-rules>

13

Exercice 13 : valider votre formulaire

localhost:8080/step13

- Valider les champs
 - Firstname : **required + min 2 lettres**
 - Lastname : **required + min 2 lettres**
 - Email : **required**
 - Phone: **required** et 10 digits ⇒ **\d{10}**
- Afficher des messages en fonction des erreurs
- Utiliser la classe **md-input-invalid** sur chaque container en cas d'erreur
- Désactiver le bouton de validation si le formulaire n'est pas valide

SOLUTION

localhost:8080/step13_solution

Model driven form

- Ajouter la dépendance **vuelidate**

```
npm install vuelidate --save
```

- Ajouter vuelidate au projet

```
import Vue from 'vue';
import Vuelidate from 'vuelidate';

Vue.use(Vuelidate);
```

Dans le template HTML

- Indiquer l'événement qui déclenche les changements d'états
 - appel de \$touch pour passer en dirty
 - appel de \$reset pour passer en pristine
- v-model pour le databinding

```
<div class="column is-12">  
  <label class="label" for="firstname">Firstname</label>  
  <p>  
    <input @input="$v.person.firstname.$touch()" name="firstname" v-model="person.firstname">  
  </p>  
</div>
```

Dans le composant

- dans la propriété **validations**
- modèle du formulaire
- Chaque champs est une clé
 - avec ces contraintes
 - sameAs('otherField')

```
validations: {  
    person:{  
        firstname: {  
            required,  
            minLength: minLength(2)  
        },  
        lastname: {  
            required,  
            minLength: minLength(2)  
        },  
        email: {  
            required,  
            email  
        }  
    }  
}
```

Gestion des erreurs

- Sur le propriété `$v`: accès à chaque champs
 - objet `$error`
 - map de contraintes
 - `required`
 - `minlength...`

```
<p :class="{ 'md-input-invalid': $v.person.firstname.$error }">  
  <input @input="$v.person.firstname.$touch()" name="firstname" v-model="person.firstname">  
  <span v-show="!$v.person.firstname.required">Champs obligatoire</span>  
</p>
```

Les validateurs

- Doivent être importés
 - required
 - minLength, maxLength
 - between
 - url
 - ...

<https://vuelidate.netlify.com/#sub-builtin-validation>

Exercice 14 : valider votre formulaire

localhost:8080/step14

- Transformer **Form** en mode **Model-Driven**
- Utiliser la librairie vuelidate
- Tenez compte du mode “création” aussi
- Attention pas de pattern pour le téléphone

SOLUTION

localhost:8080/step14_solution

Vuelidate

Validation personnalisée

Créer ses propres validateurs

- Créer sa fonction de validation
 - Si OK : retourne **true**
 - Si KO: retourne **false**
- On peut lui passer des paramètres

```
export default {  
    myValidator(args) {  
        return (value) => {  
            //return true or false  
        }  
    }  
}
```

Utilisation du validateur

- Dans les règles de validation

```
import customValidator from './customValidator.js'
```

```
...  
validations: {  
    person:{  
        phone: {  
            pattern:customValidator.myValidator(1,2,3)  
        }  
    }  
}
```

15

Exercice 15 : votre validateur

localhost:8080/step15

- Créer une classe **customValidators** dans **components/form**
- Créer un validateur (méthode statique) qui vérifie le format du téléphone
- Créer un validateur (méthode statique) qui vérifie l'adresse e-mail
 - respectant le format "**nom.p@sfeir.com**"
- Associer ce validateur au control "mail"
- Afficher dans le HTML le message correspondant au type d'erreur

SOLUTION

localhost:8080/step15_solution

Les Filtres

Exemples

hello

```
{ { "hello" | uppercase } }
```

HELLO

Exemples

1368730200

```
{ { "1368730200" | date } }
```

Jeudi 16 mai 2013 20H50

Exemples

1234.562342

```
{ { "1234.562342" | currency } }
```

1 234,56 €

Transformer les données avant de les afficher

Syntaxe

- A la suite d'une expression

```
{ { expression | filter1 } }
```

- On peut les chaîner

```
{ { expression | filter1 | filter2 } }
```

- On peut leur passer des paramètres

```
{ { expression | filter1(param1,param2) } }
```

Utilisation

- Dans la propriété filter du composant
- le nom de la clé sera le nom dans le template
- Une méthode qui prend une valeur en argument et en retourne une nouvelle

```
filters: {  
  capitalize: function (value) {  
    if (!value) return ""  
    value = value.toString()  
    return value.charAt(0).toUpperCase() + value.slice(1)  
  }  
}
```

16

Exercice 16 : Je veux des N/A

localhost:8080/step16

- Créer un filtre **NaFilter** dans un répertoire filter
- Afficher "N/A" s'il n'y a pas de manager ou d'entité associés

SOLUTION

localhost:8080/step16_solution

Les modifiers

Les modifiers

- pour customiser une directive
- peuvent être chaînés
- après un .

```
<form v-on:submit.prevent="onSubmit">  
  <input v-model.trim="msg">  
</form>
```

Pour les event

- .stop
- .prevent
- .once
- .left
- .right
- .middle
- ...

<https://vuejs.org/v2/guide/events.html#Event-Modifiers>

Le clavier...

- .{keyCode | keyAlias}
 - .enter, .tab, .delete, .esc, .space, .up, .down, .left, .right
 - .ctrl, .alt, .shift, .meta
- On peut définir des alias
 - Vue.config.keyCodes.f1 = 112

```
<input v-on:keyup.13="submit">  
<input v-on:keyup.enter="submit">  
<input @keyup.enter="submit">
```

<https://vuejs.org/v2/guide/events.html#Key-Modifiers>

Pour le v-model

- **.lazy** : écoute les event change et non input
- **.number**: pourcaster en nombre
- **.trim**

Exercice 17 : On filtre

17

localhost:8080/step17

- Nous avons ajouté un composant **SearchBar**
- Dans le composant People, vous avez une méthode **filterPeople**
- Utilisons un modifier pour déclencher le filtre en tapant sur entrée
- on propage un événement custom **search** que l'on écoute dans le people

SOLUTION

localhost:8080/step17_solution

Les directives

directives natives

- v-bind
- v-for
- v-show
- v-model
- v-if
- v-on
- v-once
- ...

directive custom

- Souvent le composant suffit
- mais pour enrichir le comportement d'une élément natif
- s'enregistre de façon globale ou localement sur un composant
- le nom sera préfixé par **v-**
- Peut passer une simple fonction=bind/update

Vue.directive(**foo**, {})

OU

directives:{

foo: {}

}

<input **v-foo**>

Cycle de vie

- **bind** une fois, quand on est attaché à l'élément
- **inserted**: on est inséré dans le parent
- **update**: le composant a été mis à jour
- **componentUpdated**: après que le composant **et ces enfants** aient été modifiés
- **unbind**: la directive est détaché de l'élément

```
directives:{  
  focus: {  
    inserted: function (el) {  
      // Focus the element  
      el.focus()  
    }  
  }  
}
```

Arguments

- el: élément sur laquelle la directive est attachée
- binding: map avec
 - name nom sans le v-
 - value
 - expression
 - arg
 - modifiers
- vnode
- oldVnode (pour update et updateComponent)

18

Exercice 18 : Tout en majuscule

localhost:8080/step18

- Créez une directive **v-upper** qui permet de modifier la valeur saisie dans un champs pour la mettre en majuscule
- Utilisez la sur le champs de filtre

SOLUTION

localhost:8080/step18_solution

BONUS

Communication avancée

3 façons de communiquer

- entre parent- enfant: event custom
- entre autres éléments
 - bus de communication
 - architecture flux

Un simple bus

- Créer une instance de Vues dédiée
- l'importer dans les différents composants
- Utiliser ces méthodes
 - \$on
 - \$emit

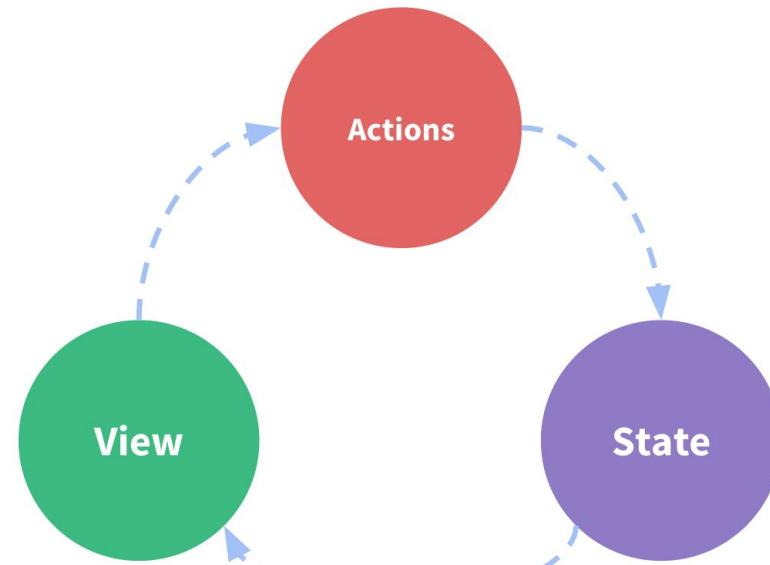
```
import Vue from 'vue'  
var bus = new Vue({})  
export default bus;
```

```
import bus from '../path/toBus';  
bus.$on(xxx,function(datas){...});  
bus.$emit(xxx,datas);
```

VUEX

- Architecture dataflow: like Redux :)
- librairie tiers **MAIS** officielle
 - à installer

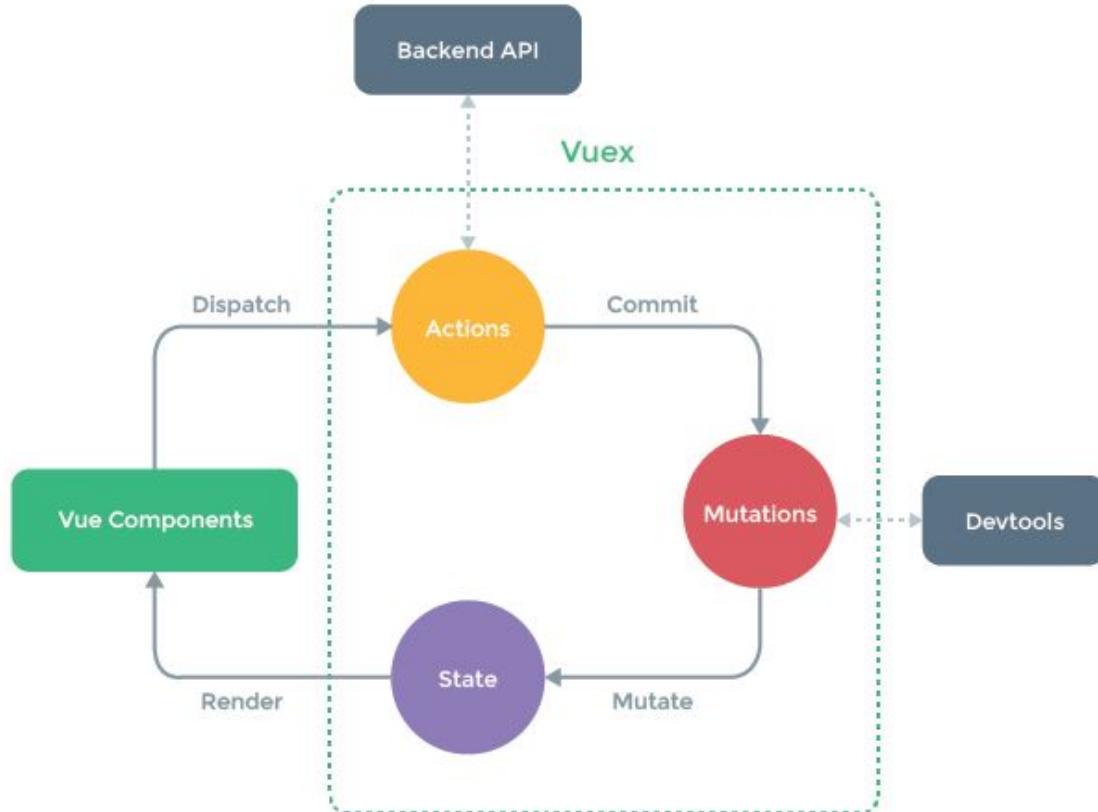
```
npm install vuex --save
```



VUEX

- 3 éléments

- state et getters
- mutations
- actions



Le Store

- unité de base qui va contenir
 - le state
 - les getters
 - les actions
 - les mutations
- va être injecté dans notre instance Vue:
this.\$store dans les composants

```
new Vue({  
  ...  
  store  
})
```

```
import Vue from 'vue'  
import Vuex from 'vuex'  
Vue.use(Vuex)  
// initial state  
const state = {...}  
// getters  
const getters = {...}  
// actions  
const actions = {...}  
// mutations  
const mutations = {...}  
export default new Vuex.Store({  
  state,  
  actions,  
  getters,  
  mutations  
})
```

Le State

- map de clé valeurs
- Propriété **state**
- récupérer dans le store par `this.$store`

Dans le Store:

```
const state = {  
    people: [],  
    search: ''  
}
```

Dans le composant

```
computed: {  
    search: function(){  
        return this.$store.state.search;  
    }  
},
```

Les getters

- Pour travailler sur le state
- Propriété **getters**
- “computed” du store
- Pour centraliser un traitement
 - filtre, tri...

Dans le Store:

```
getters: {  
    doneTodos : state => {  
        return state.todos.filter(todo => todo.done)  
    }  
}
```

Dans le composant

```
computed: {  
    doneTodosCount () {  
        return this.$store.getters.doneTodos  
    }  
},
```

Les mutations

- **SEULE** façon de modifier le state
- Propriété **mutations**
- déclenchée par un **commit**
- les mutations sont **réactives**
- les mutations sont **synchrones**
- Utiliser des constantes pour les types de mutations

Dans le Store:

```
mutations: {  
    increment (state,payload) {  
        state.count=payload.amount;  
    },  
    [types.FETCH_ALL](state, { people }) {  
        state.people = people;  
    }  
}
```

Dans le composant

```
this.$store.commit('increment', 10)  
this.$store.commit('increment', {amount:10})  
this.$store.commit({type:'increment',amount:10})
```

Les actions

- Propriété **actions**
- commit des mutations
- traitement **asynchrones**
- déclenchée par un **dispatch**

Dans le Store:

```
actions: {  
  increment (context) {  
    context.commit('increment')  
  }  
}
```

Dans le composant

```
this.$store.dispatch('increment', 10)  
this.$store.dispatch('increment', {amount:10})  
this.$store.dispatch({type:'increment', amount:10})  
,
```

Les helpers

- mapGetter
- mapMutations
- mapActions

Dans le composant

```
computed: {  
  ...mapGetters([  
    'doneTodosCount'  
)  
,  
methods: {  
  ...mapMutations(['increment']),  
  ...mapActions({add: 'increment'})  
}
```

19

Exercice 19 : Je branche Vuex

localhost:8080/step19

- Vous disposez d'un squelette de Store
- Nous allons mettre en place une architecture Vuex pour
 - récupérer la liste des utilisateurs
 - Filtrer le tableau
- Une action pour la requête
- Une mutation et un getter pour le filtre

SOLUTION

localhost:8080/step19_solution

BONUS

Tests unitaires

Pourquoi des tests unitaires alors que...



Parce que...



Les tests unitaires permettent de...

- éviter les régressions
- documenter le code
- s'assurer qu'un bug ne réapparaîsse pas / plus
- challenger l'architecture de l'appli
- refuser une PR de Cyril qui n'a pas voulu les écrire ;-)

Tests unitaires dans une appli Vue

On teste :

- le cycle de vie du composant
- les méthodes
- les props
- l'affichage / rendu du composant

Pour cela, il suffit d'importer Vue et votre composant dans un test unitaire

Dans le test:

```
import Vue from 'vue'  
import MyComponent from 'path/to/MyComponent.vue'  
  
describe('MyComponent', () => {  
  it('has a created hook', () => {  
    expect(typeof MyComponent.created).toBe('function')  
  })  
  it('correctly sets the message when created', () => {  
    const vm = new Vue(MyComponent).$mount()  
    expect(vm.message).toBe('bye!')  
  })  
  it('sets the correct default data', () => {  
    expect(typeof MyComponent.data).toBe('function')  
    const defaultData = MyComponent.data()  
    expect(defaultData.message).toBe('hello!')  
  })  
})
```

Quelques notes sur les tests unitaires

- De nombreuses équipes "choisissent" (ou oublient) de tester unitairement les applis Front : *c'est une mauvaise pratique car la logique applicative se déplace progressivement chez le client au lieu d'être prise en charge par un serveur*
- Certains font le choix de ne pas tester le rendu (seule la logique est testée unitairement) : *c'est tolérable, mais prévoyez alors des tests end-to-end pour votre appli ([vue-cli propose cypress.io](#) qui est très facile d'utilisation)*
- Si vous vous retrouvez à écrire plusieurs fois le même test pour des composants différents, réécrivez vos composants ! *Les tests unitaires sont de bons révélateurs des infractions au principe DRY (Don't Repeat Yourself)*
- Attention au contexte des fonctions de votre librairie de test ! *La plupart des frameworks de testing (jasmine, mocha, ...) utilisent des méthodes comme describe(...), beforeEach(...) ou it(...) et sont exécutées une fois l'interprétation de tous les scripts terminée, ce qui peut avoir des effets de bord inattendus !*

20

Exercice 20 : J'écris des tests unitaires

localhost:8080/step20

- Vous disposez d'un exemple dans le dossier tests/unit
- Créez le fichier components/SearchBar.spec.js dans le dossier ???
- Ajoutez les tests unitaires de ce composant, qui doit :
 - Récupérer la valeur entrée dans la barre de recherche
 - Transmettre au composant parent la valeur recherchée

Ressources

Ressources

- Site Officiel : <https://vuejs.org/>
- vue-cli: <https://morningstar.engineering/vue-ui-a-first-look-916600d9a918>
- Extensions:
<https://chrome.google.com/webstore/detail/vuejs-devtools/nhdogjmejiglipccpnnaanhbledajbpd>
- vue-router : <https://router.vuejs.org/en/>
- vuelidate : <https://monterail.github.io/vuelidate/>
- vee-validate: <http://vee-validate.logaretm.com/#about>
- Ressources: <https://github.com/vuejs/awesome-vue>

Vue.js



Guide API Examples Ecosystem Translate



The Progressive
JavaScript Framework

GET STARTED

GITHUB

Approachable

Already know HTML, CSS and
JavaScript? Read the guide and start
building things in no time!

Versatile

Simple, minimal core with an
incrementally adoptable stack that
can handle apps of any scale.

Performant

18kb min+gzip Runtime
Blazing Fast Virtual DOM
Minimal Optimization Efforts