Vue.js 2

Table des matières

1. Installation

```
npm i vue -g
```

2. Création de projet

a. Projet avec CDN

Utile pour faire des démos

```
<div id="app"></div>
<script src="https://unpkg.com/vue"></script>
```

b. Avec Vue-CLI

Documentation

Installation de Vue-CLI en global

```
npm install -g vue-cli
```

Templates :

- **Webpack** (VueRouter, eslint, tests avec Karma + Mocha, e2e) (Github)

```
vue init webpack <project-name>
```

Puis « **npm i** »

- Simple (page simple avec CDN)

```
vue init simple <project-name>
```

- Browserify
- Etc.

NPM Scripts :

En développement (hot reloading)

```
npm run dev
```

Tests

```
npm run test
```

Ou spécifiquement

```
npm run unit
npm run e2e
```

Lint

```
npm run lint
```

Build

```
npm run build
```

3. EDI

- **VS Code**. Extensions : vetur (highligth,intelliSense, etc.), snippets, vscode-icons
- **WebStorm**

4. VueRouter

Github, documentation, simple routing

Si le router n'est pas installé

```
npm i vue-router -S
```

main.js

```
import Vue from 'vue'
import App from './App'
import VueRouter from 'vue-router';
import { routes } from './routes';

Vue.config.productionTip = false

Vue.use(VueRouter);

const router = new VueRouter({
  routes
});

/* eslint-disable no-new */
new Vue({
  el: '#app',
  router,
  template: '<App/>',
  components: { App }
})
```

Création des **routes** dans un fichier « routes.js »

```
import Home from './components/Home';
import PostList from './components/posts/PostList';
import PostDetail from './components/posts/PostDetail';

export const routes = [
    { path: '/', component: Home },
    { path: '/posts', component: PostList },
    { path: '/posts/:id', component: PostDetail },
    { path: '*', redirect: '/' }
];
```

Ajouter dans App.vue le conteneur pour les vues

```
<router-view></router-view>
```

a. Mode

Par défaut « hash ». Changer le mode pour html5 **history** (sans # dans url):

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

b. Router link

```
<nav>
    <router-link to="/">Home</router-link>
    <router-link to="/posts">Post list</router-link>
    <router-link to="/posts/10">Post detail</router-link>
</nav>
```

Plus query et fragment

```
 <router-
link :to="{path:'/posts/10', query:{q:'mysearch'},hash:'#section1'}">Post de
tail</router-link>
```

⇨ http://localhost:8080/posts/10?q=mysearch#section1

Active class

L'attribut **active-class** avec la **classe css** à ajouter quand la route active correspond + attribut **exact** pour éviter qu'un lien reste actif (avec children)

```
<router-link to="/" active-class="active" exact>Home</router-link>
```

CSS

```
.active{
  color:orange;
}
```

c. Récupération de paramètre passé (avec this.$route)

```
<template>
<div>
    <h1>Post</h1>
    <p>Id: {{ id }}</p>
</div>
</template>
<script>
export default {
    data(){
        return {
            id:this.$route.params.id
        }
    }
}
</script>
```

Récupération de **query** et **fragment**

```
<script>
export default {
    data(){
        return {
            id:this.$route.params.id,
            q:this.$route.query.q,
            fragment: this.$route.hash
        }
    }
}
</script>
```

d. Navigation par programmation (avec this.$router)

```
<template>
<div>
    <h1>Home</h1>
    <button @click="goPostDetail">Go post id 10</button>
</div>
</template>

<script>
export default {
    methods:{
        goPostDetail(){
            this.$router.push('/posts/10');
        }
    }
}
</script>
```

Avec params + query + fragment

```
this.$router.push({path:'/posts/50',query:{ q:'mysearch' }, hash:'#section1'
});
```

Redirect

On peut rediriger vers la page d'accueil par exemple. On peut également utiliser redirect sur n'importe quelle route (exemple sur « home » pour rediriger vers « posts »)

```
export const routes = [
    { path: '/', component: Home },
    /* etc. */
    { path: '*', redirect: '/' }
];
```

e. Named route

```
export const routes = [
    { path: '/', component: Home },
    { path: '/posts', component: PostList,  name: 'postList' },
    { path: '/posts/:id', component: PostDetail, name:'postDetail' }
];
```

Lien (avec « **:to** »)

```
<router-link :to="{name:'postList'}">Post list</router-link>
```

Avec params

```
<router-link :to="{name:'postDetail', params:{id: 20}}">Post detail</router-
link>
```

Avec params + query + fragment

```
<router-
link :to="{name:'postDetail', params:{id: 20}, query:{q:'mysearch'},hash:'#s
ection1'}">Post detail</router-link>
```

Navigation par programmation

```
this.$router.push({ name: 'postList'});
```

Avec params

```
this.$router.push({ name:'postDetail', params:{id:50}});
```

Avec params + query + fragment

```
this.$router.push({ name:'postDetail', params:{id:50}, query:{ q:'mysearch'
}, hash:'#section1'});
```

f. Children

On crée un component « Post.vue » par exemple

```
<template>
<div>
    <h1>Posts</h1>
    <hr />
    <router-view></router-view>
</div>
</template>
<script>
export default {}
</script>
```

Modification des routes

```
export const routes = [
    { path: '/', component: Home },
    {
        path: '/posts', component: Post, children: [
            { path: '', component: PostList, name: 'postList' },
            { path: ':id', component: PostDetail, name: 'postDetail' }
        ]
    },
    { path: '*', redirect: '/' }
];
```

On peut désormais switcher entre « Home » et « Post ». De plus « PostList » et » PostDetail » seront affichés dans le router-view de « Post »

g. Named views

Exemple

Header.vue

```
<template>
    <h1>My Header</h1>
</template>
```

Home.vue

```
<template>
    <h1>Home</h1>
</template>
```

App.vue

```
<template>
  <div class="container">
    <router-view name="header"></router-view>
    <router-view></router-view>
  </div>
</template>
```

« header »

Default view

Route

```
export const routes = [
    {
        path: '/', components: {
            default: Home,
            header:Header
        }
    },
    /* etc. */
];
```

h. Transitions

Exemple animation d'opacité

Entourer le router d'une transition

```
<transition name="fade" mode="out-in">
   <router-view></router-view>
</transition>
```

CSS

```
.fade-enter-active, .fade-leave-active {
  transition: opacity .5s ease;
}
.fade-enter, .fade-leave-active {
  opacity: 0
}
```

i. Guards

Documentation

j. Before each et after each (pour toutes les routes)

```
router.beforeEach((to, from, next) => {
  console.log('before each', to, from);
  next();
});
router.afterEach((to, from) => {
  console.log('after each', to, from);
});
```

Il est possible de faire une redirection

beforeEnter (Activation) sur la **route**

```
export const routes = [
    { path: '/', component: Home },
    {
        path: '/posts', component: Post, children: [
            { path: '', component: PostList, name: 'postList' },
            {
                path: ':id', component: PostDetail, name: 'postDetail',
                beforeEnter: (to, from, next) => {
                    let result = confirm('Activate post detail ?');
                    next(result);
                }
            }
        ]
    }
];
```

Sur le **component**

⇨ beforeRouteEnter et beforeRouteLeave

```
<script>
export default {
    beforeRouteEnter: (to, from, next) => {
        console.log('before route enter guard');
        next();
    },
    beforeRouteLeave: (to, from, next) => {
        console.log('before route leave guard');
        next();
    }
}
</script>
```

5. Binding

a. Attribut

Documentation

« **v-bind :** » suivi de l'**attribut** ou « **:** » suivi de l'**attribut**

Exemples :

« **v-bind :href** » ou « **:href** »

v-html

Par défaut une string interpolation => affiche « My content »

:class

Class et style binding

```
<p :class="'red'">{{ myContent }}</p>
```

CSS

```
.red {
      color: red;
    }
```

Autre exemple

```
<div id="app"><p :class="{red:isRed}">{{ myContent }}</p></div>
  <script>
    new Vue({
      el: '#app',
      data: {
        isRed: true,
        myContent: 'My content'
      }
    })
  </script>
```

:style

```
  <p :style="{backgroundColor:'red'}">{{ myContent }}</p>
```

Avec **v-html** => affiche le **texte en italique**

```
<span v-html="myHtmlContent"></span>
```

```
 new Vue({
      el: '#app',
      data: {
        myHtmlContent: '<i>My content</i>',
      }
})
```

v-once

Render une seule fois

```
<div id="app">
    <span v-once>{{ myContent }}</span>
    <button @click="change">Change</button>
  </div>
  <script>
    new Vue({
      el: '#app',
      data: {
        myContent: 'My content',
      },
      methods: {
        change() {
          this.myContent = 'New content';
```

Après un clic sur le bouton le content ne sera pas changé avec v-once

```
        }
      }
    })
  </script>
```

b. Events

Documentation

« **v-on :** » suivi de l'**event** ou « **@** » suivi de l'**event**

Exemples : « **v-on :click** » ou « **@click** »

```
<div id="app">
    <button v-on:click="onClick">Clic!</button>
  </div>

  <script>
    new Vue({
      el: '#app',
      methods: {
        onClick(event) {

        }
      }
    })
  </script>
```

Passage de paramètre et $event

```
<div id="app">
    <button v-on:click="onClick(10,$event)">Clic!</button>
  </div>

  <script>
    new Vue({
      el: '#app',
      methods: {
        onClick(p1,event) {

        }
      }
    })
  </script>
```

Modifiers

Documentation

Exemple

```
!-- the submit event will no longer reload the page -->
  <form v-on:submit.prevent="onSubmit"></form>
```

Autre exemple : La function n'est appelée qu'avec si la touche « enter »

```
<div id="app">
    <input type="text" v-on:keyup.enter="onSubmit" />
  </div>

  <script>
    new Vue({
      el: '#app',
      methods: {
        onSubmit(event) {
          console.log(event);
        }
      }
    })
  </script>
```

c. Two way binding (v-model)

Documentation

Avec la directive « **v-model** »

```
  <div id="app">
    <input type="text" v-model="username" />
    {{ username }}
  </div>

  <script>
    new Vue({
      el: '#app',
      data() {
        return {
          username: 'Marie'
        }
      }
    })
  </script>
```

d. Computed

Documentation

Exemple

```
<div id="app">
   <button @click="increment">Increment</button>
   <p>Result:{{ myComputed }}</p>
 </div>

 <script>
   new Vue({
     el: '#app',
     data: {
       count: 0
     },
     methods: {
       increment() {
         this.count++;
       }
     },
     computed: {
       myComputed: function () {
         return this.count > 5 ? 'Greater than 5' : 'Smaller than 5';
       }
     }
   })
 </script>
```

e. Conditional

Documentation

v-if

Element supprimé du DOM

```
<div id="app">
   <button @click="onToggleShow">Toggle show</button>
   <p v-if="showContent">{{ myContent }}</p>
 </div>
 <script>
   new Vue({
     el: '#app',
     data: {
       showContent: false,
       myContent: 'My content'
     },
     methods: {
       onToggleShow() {
         this.showContent = !this.showContent;
       }
     }
   })
 </script>
```

+v-else, v-else-if

v-show

La différence, l'élément passe à « display :none »

```
<p v-show="showContent">{{ myContent }}</p>
```

```
<button>Toggle show</button>
<p style="display: none;">My content</p> == $0
```

f. v-for

Documentation

```
<div id="app">
  <ul>
    <li v-for="item in items">{{ item }}</li>
  </ul>
</div>

<script>
  new Vue({
    el: '#app',
    data: {
      items:['item 1', 'item 2', 'item 3']
    }
  })
</script>
```

Avec **index**

```
<ul><li v-for="(item,index) in items"> {{ index }} - {{ item }}</li></ul>
```

:key et push

```
<div id="app">
  <ul>
    <li v-
for="(item,index) in items" :key="item"> {{ index }} - {{ item }}</li>
  </ul>
  <button @click="onPush">Add New Item</button>
</div>
<script>
  new Vue({
    el: '#app',
    data: {
      items: ['item 1', 'item 2', 'item 3']
    },
    methods: {
      onPush() {
        this.items.push('item ' + this.items.length);
      }
    }
  })
</script>
```

6. Filters

Documentation

```
<div id="app"><p>{{ title | toUpperCase }}</p></div>
    <script src="https://unpkg.com/vue"></script>
    <script>
        new Vue({
            el: "#app",
            data: {
                title: "My title"
            },
            filters: {
                toUpperCase: function (value) {
                    return value.toUpperCase();
                }
            }
        });
    </script>
```

On peut enregistrer le filtre de manière globale de manière **globale** à le rendre disponible à toute l'application

```
Vue.filter('toUpperCase',function(value){
     return value.toUpperCase();
});
```

Il est possible de **chainer les filtres**.

```
<p>{{ title | filter1 | filter2 }}</p>
```

Filtrer une liste avec un computed

```
<div id="app">
        <input type="text" v-model="filterText">
        <ul><li v-for="fruit in filteredFruits">{{ fruit }}</li></ul>
    </div>
    <script src="https://unpkg.com/vue"></script>
    <script>
        new Vue({
            el: "#app",
            data: {
                fruits: ['Banane', 'Pomme', 'Poire'],
                filterText:''
            },
            computed: {
                filteredFruits: function () {
                    return this.fruits.filter((fruit) => {
                        return fruit.match(this.filterText);
                    });
                }
            }
        });
    </script>
```

7. Mixins

Documentation

Offrent une autre manière de réutiliser le code

```
// mixin
var myMixin = {
    created: function () {
        this.hello()
    },
    methods: {
        hello: function () {
            console.log('hello from mixin!')
        }
    }
}

// a component that uses the mixin
var Component = Vue.extend({
    mixins: [myMixin]
})

var component = new Component() // -> "hello from mixin!"
```

Création d'un morceau de code que l'on pourra importer/ réutiliser dans des components

Créer une mixin « **globale** »

```
Vue.mixin({
    created: function () {
        console.log('My global mxin')
    }
});

new Vue({
    el: "#app"
});
```

8. Custom directives

Documentation

Sans paramètre

```
 <div id="app">
       <p v-highlight>My text</p>
    </div>
    <script src="https://unpkg.com/vue"></script>
    <script>
        new Vue({
            el: "#app",
            directives: {
                highlight: {
                    bind(el, binding, vnode) {
                        el.style.backgroundColor = 'red';
                    }
```

```
                }
            }
        });
    </script>
```

a. Avec un paramètre

```
<div id="app">
      <p v-highlight="'red'">My text</p>
    </div>
    <script src="https://unpkg.com/vue"></script>
    <script>
        new Vue({
            el: "#app",
            directives: {
                highlight: {
                    bind(el, binding, vnode) {
                        el.style.backgroundColor = binding.value;
                    }
                }
            }
        });
    </script>
```

Autre façon de faire avec « arguments »

```
<div id="app">
        <p v-highlight:background="'red'">My text</p>
    </div>
    <script src="https://unpkg.com/vue"></script>
    <script>
        new Vue({
            el: "#app",
            directives: {
                highlight: {
                    bind(el, binding, vnode) {
                        if (binding.arg === 'background') {
                            el.style.backgroundColor = binding.value;
                        }
                    }
                }
            }
        });
    </script>
```

b. Avec plusieurs paramètres

```
 <div id="app">
       <p v-
highlight="{ backgroundColor : 'red', color:'white' }">My text</p>
    </div>
    <script src="https://unpkg.com/vue"></script>
    <script>
        new Vue({
            el: "#app",
            directives: {
                highlight: {
                    bind(el, binding, vnode) {
                        el.style.backgroundColor = binding.value.backgroundC
olor;
                        el.style.color = binding.value.color;
                    }
                }
            }
        });
    </script>
```

c. Enregistrer une directive de manière « globale »

```
Vue.directive('highlight', {
    bind(el, binding, vnode) {
        el.style.backgroundColor = binding.value;
    }
});
new Vue({
    el: "#app"
});
```

9. Form validation

Il faut actuellement se tourner vers vue-validator ou vee-validate

10. Vue instance

Documentation

11. Lifecycle

Documentation

- beforeCreate
- created
- beforeMount
- mounted
- beforeUpdate
- updated
- beforeDestroy
- destroyed

```
 <div id="app"><h1>{{ title }}</h1><button @click="title = 'New title'">Upda
te Title</button><button @click="destroy">Destroy</button></div>
  <script>
    new Vue({
      el: '#app',
      data: {
        title: "My title"
      },
      beforeCreate: function () {
        console.log('beforeCreate');
      },
      created: function () {
        console.log('created');
      },
      beforeMount: function () {
        console.log('beforeMount');
      },
      mounted: function () {
        console.log('mounted');
      },
      beforeUpdate: function () {
        console.log('beforeUpdate');
      },
      updated: function () {
        console.log('updated');
      },
      beforeDestroy: function () {
        console.log('beforeDestroy');
      },
      destroyed: function () {
        console.log('destroyed');
      },
      methods: {
        destroy: function () {
          this.$destroy();
        }
      }
    });
  </script>
```

12. Components

Documentation

```
<div id="app">
   <my-component></my-component>
 </div>

 <script>
   Vue.component('my-component', {
     template: `
     <div>
           <h1>{{ title }}</h1>
           <button @click="onChangeTitle">Change title</button>
     </div>
     `,
     data() {
       return {
         title: 'My component title'
       }
     },
     methods: {
       onChangeTitle() {
         this.title = 'New title';
       }
     }
   });
   new Vue({
     el: '#app'
   })
 </script>
```

a. Component «.vue »

```
<template>
 <div>
    <h1>{{ title }}</h1>
    <button @click="onChangeTitle">Change title</button>
  </div>
</template>
<script>
export default {
  data () {
    return {
      title: 'My component title'
    }
  },
  methods: {
    onChangeTitle () {
      this.title = 'New title'
    }
```

```
  }
}
</script>
```

Pour utiliser le component, l'importer dans « App.vue » par exemple

```
<template>
  <div id="app">
    <my-component></my-component>
  </div>
</template>

<script>
import MyComponent from './components/MyComponent'

export default {
  name: 'app',
  components: {
    MyComponent
  }
}
</script>
```

b. Dynamic component

Documentation

Pouvoir switcher de component

c. Communication Parent > child (props)

```
<template>
 <div>
    <h1>Parent</h1>
    <child :myName="'my value!'"></child>
  </div>
</template>

<script>
import Child from './child';

export default {
    components:{
        Child
    }
}
</script>
```

Ou par rapport à une propriété

```
<template>
 <div>
    <h1>Parent</h1>
    <child :myName="'my value!'"></child>
  </div>
```

```
</template>

<script>
import Child from './child';

export default {
    data(){
        return {
        name:'my value'
        }
    },
    components:{
        Child
    }
}
</script>
```

Child

Récupération en props

```
<template>
 <div>
    <p>{{ myName }}</p>
  </div>
</template>

<script>
export default {
  props: ['myName']
}
</script>
```

Définir le type

```
export default {
  props:{
    myName:{
      type:String,
      required:true
    }
  }
}
```

Définir une valeur par défaut

```
export default {
  props:{
    myName:{
      type:String,
      default:'Marie'
    }
  }
}
```

d. Child >Parent ($emit)

Child

Exemple au clic sur un button

```
methods:{
    onSend(){
      this.$emit('onSend','my send value');
    }
  }
```

Parent

```
<template>
 <div>
    <h1>Parent</h1>
    <child @onSend="onNotified"></child>
  </div>
</template>

<script>
import Child from './child';

export default {
    data(){
        return {
        name:'my value'
        }
    },
    methods:{
        onNotified(event){
            console.log('notified',event); // event = 'my send value'
        }
    },
    components:{
        Child
    }
}
</script>
```

Autre possibilité pour la communication entre components => event bus

13. Animations / transitions

Documentation

14. Vue resource

Github

Installation

```
npm install vue-resource
```

Ou CDN

```
<script src="https://cdn.jsdelivr.net/vue.resource/1.2.1/vue-
resource.min.js"></script>
```

GET

```
Vue.use(VueResource);

new Vue({
    el: "#app",
    data: {
        posts: []
    },
    methods: {
        getPosts: function () {
            this.$http.get('https://jsonplaceholder.typicode.com/posts')
            .then((response) => {
                return response.json();
            }).then((data) => {
                this.posts = data;
            });
        }
    }
});
```

POST

```
 <div id="app">
        <button @click="getPosts">Load posts</button>
        <div>
            <form v-on:submit.prevent="onSubmit">
                <div class="form-group">
                    <label>Title</label>
                    <input class="form-control" type="text" v-
model="post.title">
                </div>
                <div class="form-group">
                    <input class="form-control" type="text" v-
model="post.body">
                </div>
                <button class="btn btn-primary">Submit</button>
            </form>
        </div>
    </div>
```

```
    <script src="https://cdn.jsdelivr.net/vue.resource/1.2.1/vue-
resource.min.js"></script>
    <script src="https://unpkg.com/vue"></script>
    <script>
        Vue.use(VueResource);

        new Vue({
            el: "#app",
            data: {
                post: {
                    title: '',
                    body: ''
                }
            },
            methods: {
                onSubmit() {
                    this.$http.post('https://jsonplaceholder.typicode.com/po
sts', this.post)
                        .then((response) => {
                            return response.json();
                        }, (error) => {
                            console.log('error', error);
                        }).then((post) => {
                            console.log('new post', post);
                        });
                }
            }
        });
    </script>
```

Interceptors

```
Vue.use(VueResource);

// interceptors
Vue.http.interceptors.push((request, next) => {
    console.log('intercept request', request);

    next((response) => {
        console.log('intercept response', response);
    });
});
```

Abort a request : exemples

Resource

Creating resource

Methods

- `resource(url, [params], [actions], [options])`

Default Actions

```
get: {method: 'GET'},
save: {method: 'POST'},
query: {method: 'GET'},
update: {method: 'PUT'},
remove: {method: 'DELETE'},
delete: {method: 'DELETE'}
```

```
Vue.use(VueResource);
// options
Vue.http.options.root = 'https://jsonplaceholder.typicode.com';

new Vue({
    el: "#app",
    data: {
        posts: [],
        post: {
            title: '',
            body: ''
        },
        resource: {}
    },
    methods: {
        getPosts: function () {
            this.resource.get()
                .then((response) => {
                    return response.json();
                }).then((data) => {
                    this.posts = data;
                });
        },
        onSubmit() {
            this.resource.save(this.post)
                .then((response) => {
                    return response.json();
                }, (error) => {
                    console.log('error', error);
                }).then((post) => {
                    console.log('new post', post);
                });;
        }
    },
```

```
    created() {
        this.resource = this.$resource('posts');
    }
});
```

15. Vuex

Documentation, Github

```
npm i vuex -S
```

16. Support TypeScript

Documentation