**VUE**

**Dure totale du module : 21H00**

Table des matières

[Objectifs pédagogiques : 3](#_Toc111661460)

[Notions / Concept de base de VUE 4](#_Toc111661461)

[Utilisation des computed Properties 4](#_Toc111661462)

[Exercice : Bug évènements + fonctions (interférences) 4](#_Toc111661463)

# Objectifs pédagogiques :

Être capable de comprendre les enjeux des Frameworks

Être capable d'initialiser et paramétrer un projet grâce à un Frameworks front-end

Être capable de réaliser des interface utilisateur grâce à un Frameworks front-end

~~Être capable de mettre en place un système de Routing (url)~~

Être capable de gérer des requêtes HTTP (BDD)

Être capable de connecter une application à une base de données

# Notions / Concept de base de VUE

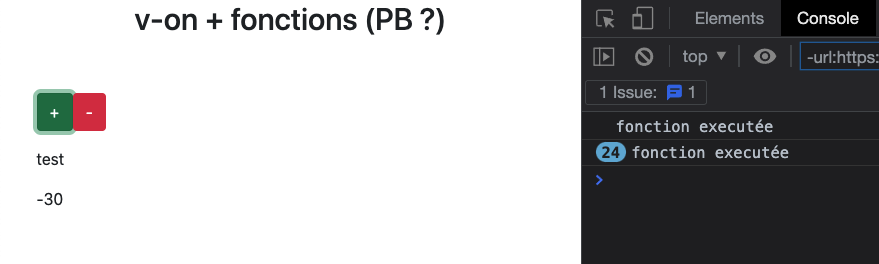
### Utilisation des computed Properties

Au cours du développement d’une application VueJS, (plus complète), nous allons gérer plusieurs fonctions, plusieurs données, on va très vite se retrouver dans des situations où sur la même page nous devons réagir à plusieurs évènements et exécuter des fonctions, cela va engendrer des interférences entre ces différentes fonctions que l’on utilise. Dans l’exercice suivant nous allons volontairement créer ce problème.

### Exercice : Bug évènements + fonctions (interférences)

Dans une application :

* Mettre en place un système de compteur
* Une fonction dans laquelle on manipule une variable userName (on exécute cette ƒ° directement dans un paragraphe côté HTML):
  + Un console log qui affiche « fonction exécutée »
  + Si userName est vide alors on retourne « test »
  + Sinon on retourne « un autre test »



Fichier HTML :

<h1 class="text-center">v-on + fonctions (PB ?)</h1>

<div id="monApp" class="container mt-4 mb-4 p-3">

<div class="mb-3">

<button v-on:click="augmenter(20)" type="button" class="btn btn-success mt-3">+</button>

<button v-on:click.left="reduire(10)" type="button" class="btn btn-danger mt-3">-</button>

</div>

<div class="mb-3">

<p>{{afficherNameUser()}}</p>

</div>

<div class="mb-3">

<p>{{leNombre}}</p>

</div>

</div>

Nous avons plusieurs fonctions qui sont exécutées sur la même page HTML

Fichier JS :

const app = Vue.createApp({

data() {

return {

leNombre:0,nameUser: ''};

},

methods: {

afficherNameUser() {

console.log('fonction exécutée');

if(this.nameUser ===''){

return 'test'

}

else{

return 'autre Test';

}

},

augmenter(num){

this.leNombre += num;

},

reduire(num){

this.leNombre -= num;

},

}

});

app.mount('#monApp');

Dans l’application il y a un problème : lorsque l’on clique sur les boutons qui interagissent avec le nombre, le console log (qui se situe pourtant dans une toute autre fonction) est déclenché.

Le problème vient du fait que nous exécutons directement dans le template la fonction afficherNameUser :

<p>{{afficherNameUser()}}</p>

Pour des raisons de performance (réactivité) Vue va en temps réel surveiller le DOM, à chaque interaction VueJs va recharger le DOM en arrière-plan (DOM Virtuel) et de fait va ré-exécuter les fonctions. (<https://la-cascade.io/comprendre-le-virtual-dom/>)

Afin de solutionner ce problème, VueJs intègre une alternative à methods pour gérer les fonctions, les computed properties.

Mise en place :   
modification côté HTML : on fait référence à la fonction (pas de parenthèse pour exécuter)(comme une property (une variable))

<p>{{afficherNameUser}}</p>

Modification côté JS : on déplace la fonction dans un objet computed (même utilisation que methods)

const app = Vue.createApp({

data() {

return {

leNombre:0,nameUser: ''};

},

methods: {

augmenter(num){

this.leNombre += num;

},

reduire(num){

this.leNombre -= num;

},

},

computed:{

afficherNameUser() {

console.log('fonction executée');

if(this.nameUser ===''){

return 'test'

}

else{

return 'autre Test';

}

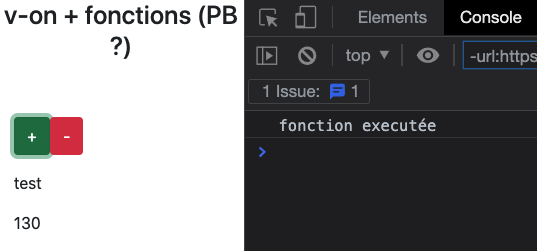
},

}

});

app.mount('#monApp');

Quand une fonction est en mode computed, VueJS va gérer les dépendances (les variables) utilisées par cette fonction, c’est-à-dire que la fonction ne sera exécutée QUE s’il y a des modifications sur les variables dont elle dépend.



Maintenant la fonction d’affichage n’est plus déclenchée par les fonctions du compteur.

[LIEN(GitHub)](https://github.com/jefff404/bookList-vue/tree/v1.9-computed-properties)