**VUE**

**Dure totale du module : 21H00**

Table des matières

[Objectifs pédagogiques : 3](#_Toc111665769)

[Notions / Concept de base de VUE 4](#_Toc111665770)

[List Rendering (Directive v-for) 4](#_Toc111665771)

[Exercice : List Rendering 5](#_Toc111665772)

[Exercice : List Rendering + index 8](#_Toc111665773)

# 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

## List Rendering (Directive v-for)

Autre directive structurelle (qui va pouvoir modifier la structure du DOM), la directive v-for, qui en combinaison avec toutes les directives vues précédemment (v-bind, v-on, v-if) va permettre d’entièrement automatiser l’affichage de Template HTML avec des données dynamiques.

La directive v-for va s’utiliser afin de répéter l’affichage d’un élément HTML (paragraphe, titre, liste, une div, tout un Template, etc…).

Exemple : Ici côté JS on aurait déjà défini une variable « maListeDeDonnées » qui serait un tableau par exemple : de la même manière que for … in ou for … of en JS, on va pouvoir définir une variable temporaire pour parcourir cette liste de données.

<p v-for="uneVarTemporaire in maListeDeDonnées">{{uneVarTemporaire}} </p>

On retrouve également la notion de scope (la portée de la variable), puisque la variable temporaire ne sera utilisable qu’à l’intérieur du v-for.

<div v-for="uneVarTemporaire in maListeDeDonnées">

<p>{{uneVarTemporaire}}</p>

</div>

## Exercice : List Rendering

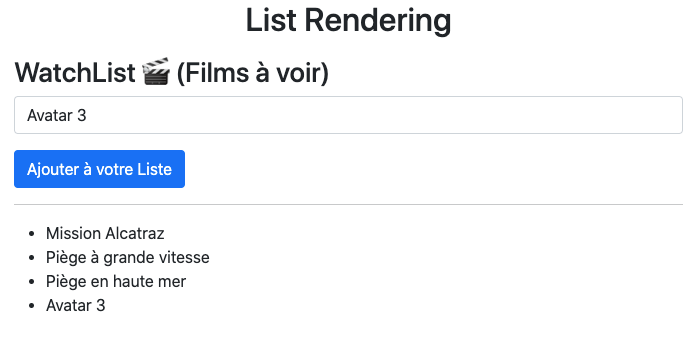
En se basant sur l’exercice précédent, mettre en place une application VueJS, dans laquelle

1 input permet d’ajouter quelque chose (le nom d’un film) à une liste (mes films à voir)

L’objectif est de répéter l’affichage des données contenues dans la liste.



On ajoute des films à la liste des films à voir,   
on répète l’affichage (ici des éléments de liste) des films que l’on ajoute



[SOLUTION(GitHub)](https://github.com/jefff404/bookList-vue/tree/v1.16-list-rendering)

Fichier JS :

const app = Vue.createApp({

data() {

return {

nomFilm:'',

mesFilms:[]

};

},

methods: {

ajouterFilm(){

this.mesFilms.push(this.nomFilm);

}

},

});

app.mount('#monApp');

Fichier HTML :

<h1 class="text-center">List Rendering</h1>

<div id="monApp">

<div class="m-3">

<h2>WatchList 🎬 (Films à voir)</h2>

<input v-model="nomFilm" type="text" class="form-control" />

<button @click="ajouterFilm()" class="btn btn-primary mt-3">

Ajouter à votre Liste

</button>

<hr />

<h2 v-if="mesFilms.length === 0">Aucun Films à Voir ?</h2>

<ul v-else>

<li v-for="unFilm in mesFilms">{{unFilm}}</li>

</ul>

</div>

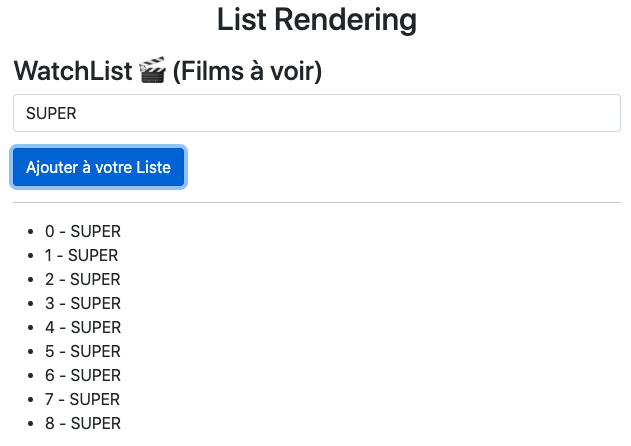
</div>

Une autre utilisation un peu différente de v-for consiste à renseigner une seconde variable temporaire pour parcourir le tableau, cela va permettre d’afficher l’index des cases du tableau parcouru.

Exemple : ici la boucle va parcourir le tableau mesFilms, et va temporairement ranger chaque case dans une variable temporaire « unFilm », avant le IN du for…in, je rajoute une seconde variable unIdex qu je peux afficher dans le paragraphe :

<li v-for="(unFilm,unIndex) in mesFilms">{{unIndex}} - {{unFilm}}</li>

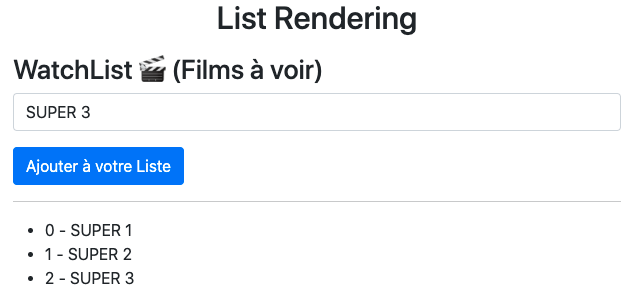
Rendu : unIndex représente l’indice de chaque case du tableau.



## Exercice : List Rendering + index

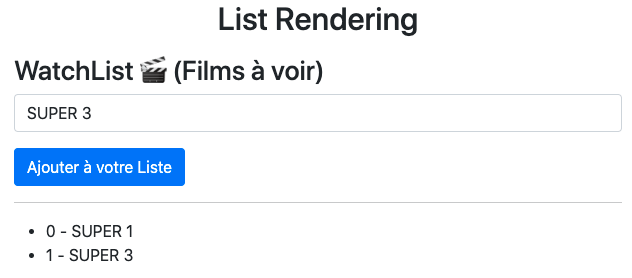
En se basant sur l’exercice précèdent (watchList), mettre en place ceci :

Quand on clique sur un des éléments rajoutés dans la liste, ça le supprime du tableau.



On a ajouté les films super 1, super 2 et super 3

En cliquant sur le film super 2, cela le supprime du tableau



[SOLUTION(GitHub)](https://github.com/jefff404/bookList-vue/tree/v1.17-list-rendering-index)

Fichier JS : on rajoute une fonction supprimer qui prend un paramètre (quel film nous voulons supprimé), dans cette fonction on utilise la fonction JS splice().

const app = Vue.createApp({

data() {

return { nomFilm:'', mesFilms:[] };

},

methods: {

ajouterFilm(){

this.mesFilms.push(this.nomFilm);

},

supprimerFilm(unIndex){

this.mesFilms.splice(unIndex,1);

}

},

});

app.mount('#monApp');

Fichier HTML : on utilise l’index de la boucle for pour le passer en paramètre de la fonction supprimer.

<div id="monApp">

<div class="m-3">

<h2>WatchList 🎬 (Films à voir)</h2>

<input v-model="nomFilm" type="text" class="form-control" />

<button @click="ajouterFilm()" class="btn btn-primary mt-3">

Ajouter à votre Liste

</button>

<hr />

<h2 v-if="mesFilms.length === 0">Aucun Films à Voir ?</h2>

<ul v-else>

<li v-for="(unFilm,unIndex) in mesFilms"

v-on:click="supprimerFilm(unIndex)">{{unIndex}} - {{unFilm}}</li>

</ul>

</div>

</div>

Une dernière méthode pour identifier précisément un élément parmi la liste des éléments qui se répète dans un v-for, consiste à faire un v-bind de l’attribut key pour le lier à la variable temporaire utilisée dans le v-for.

<li v-for="(unFilm,unIndex) in mesFilms"

v-bind:key="unFilm"

v-on:click="supprimerFilm(unIndex)">

{{unIndex}} - {{unFilm}}</li>

Cette méthode s’utilisera davantage en mettant en place une communication avec une base de données (des listes de données avec une notion de clé - valeur) comme une table SQL, ou un Objet JSON (qui ont des identifiant uniques).

Dans ce cadre ci, plutôt que de se baser sur l’index du v-for (identifiant de tableau : 0,1,2,3, etc..), nous allons préférer lier à l’attribut key des éléments qui sont répétés, l’identifiant unique de la BDD.

(Exemple d’identifiant unique généré par Firebase : -Mw2BWaG802zINFiwLsj).