**VUE**

**Dure totale du module : 21H00**

Table des matières

[Objectifs pédagogiques : 3](#_Toc111652254)

[Notions / Concept de base de VUE 4](#_Toc111652255)

[Réagir à des évènements DOM 4](#_Toc111652256)

[Exercice : setUp Application pour EventBinding 4](#_Toc111652257)

[Utilisation de la directive v-on + expressions JS 6](#_Toc111652258)

[Utilisation de la directive v-on + fonction 7](#_Toc111652259)

[Utilisation de la directive v-on + fonction avec des paramètres 8](#_Toc111652260)

[Utilisation de la directive v-on + fonction avec paramètre pour capter un évènement (Native Event Object) 9](#_Toc111652261)

# 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

## Réagir à des évènements DOM

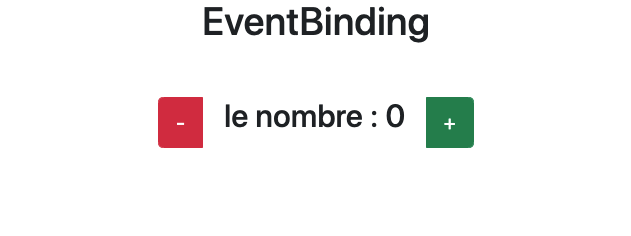
Maintenant que nous avons vu comment afficher des données via l’interpolation {{ }}, et lier des attributs HTML à des données, la prochaine étape pour construire des applications réactives, consiste à pouvoir réagir aux évènements du DOM (le click sur un bouton, le clavier, la souris de l’utilisateur…).

En Javascript (classique) pour réagir à ces évènements, on utilise la fonction addEventListener, VUE quant à lui propose la directive v-on, que l’on utilise côté HTML comme un attribut.

Afin d’illustrer l’utilisation de cette directive nous allons mettre en place une application qui augmente ou réduit un nombre.  
  
PS : la directive v-on étant une des plus utilisées on a une notation raccourcie :   
v-on :click peut s’écrire de cette façon @click

### Exercice : setUp Application pour EventBinding

Mettre en place une nouvelle application VUE dans laquelle on affiche une donnée qui est un nombre entre 2 boutons + et –



Fichier HTML :

<div id="monApp" class="container mt-4 mb-4 p-3 d-flex justify-content-center">

<div class="btn-group" role="group" aria-label="Basic mixed styles example">

<button type="button" class="btn btn-danger">-</button>

<h3 class="mx-3">le nombre : {{leNombre}}</h3>

<button type="button" class="btn btn-success">+</button>

</div>

</div>

Fichier JS :

Vue.createApp({

data(){

return {

leNombre : 0

};

},

methods: {

}

}).mount('#monApp');

Nous allons maintenant mettre en place la directive v-on directement sur les boutons pour réagir au click. (En js classique il faudrait stocker le bouton dans une variable, utiliser un AddEventListener dessus, dans ce AddEventListener décrire une fonction). L’approche de VUE est donc plus directe.

### Utilisation de la directive v-on + expressions JS

Pour mettre en place le système de compteur de l’application, nous allons placer 2 directives   
v-on :click, (après v-on : on précise le nom de l’évènement auquel on veut réagir)

Dans un v-on nous pouvons écrire des expressions JS (un calcul, une condition ternaire). Mais si le code JS se complexifie nous adopteront plutôt l’approche qui consiste à ce qu’un v-on exécute une fonction.

Exemple de v-on :click sur les boutons :

<div id="monApp" class="container mt-4 mb-4 p-3 d-flex justify-content-center">

<div class="btn-group" role="group" aria-label="Basic mixed styles example">

<button v-on:click="leNombre= leNombre-1" type="button" class="btn btn-danger">-</button>

<h3 class="mx-3">le nombre : {{leNombre}}</h3>

<button v-on:click="leNombre++" type="button" class="btn btn-success">+</button>

</div>

</div>

### Utilisation de la directive v-on + fonction

L’approche la plus commune consiste à utiliser v-on et des fonctions que l’on écrit côté JS.

Dans l’exemple ci-après nous allons créer 2 nouvelles fonctions augmenter() et réduire(), elle seront appelées par la directive v-on.

Fichier JS :

Vue.createApp({

data(){

return {

leNombre : 0

};

},

methods: {

augmenter(){

this.leNombre ++;

},

reduire(){

this.leNombre --;

}

}

}).mount('#monApp');

Fichier HTML : Nous avons le choix de faire référence à la fonction (reduire) ou d’executer la fonction (augmenter() )

<div id="monApp" class="container mt-4 mb-4 p-3 d-flex justify-content-center">

<div class="btn-group" role="group" aria-label="Basic mixed styles example">

<button v-on:click="reduire" type="button" class="btn btn-danger">-</button>

<h3 class="mx-3">le nombre : {{leNombre}}</h3>

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

</div>

</div>

### Utilisation de la directive v-on + fonction avec des paramètres

Dans cet exemple nous allons modifier nos fonctions côté JS de manière à ce qu’elles prennent en compte un paramètre.

Fichier JS :

Vue.createApp({

data(){

return {

leNombre : 0

};

},

methods: {

augmenter(num){

this.leNombre += num;

},

reduire(num){

this.leNombre -= num;

}

}

}).mount('#monApp');

Fichier HTML : on exécute les fonctions en leur passant un paramètre

<div id="monApp" class="container mt-4 mb-4 p-3 d-flex justify-content-center">

<div class="btn-group" role="group" aria-label="Basic mixed styles example">

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

<h3 class="mx-3">le nombre : {{leNombre}}</h3>

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

</div>

</div>

### Utilisation de la directive v-on + fonction avec paramètre pour capter un évènement (Native Event Object)

Nous avons vu comment réagir à un évènement (le click sur un bouton), mais dans certains cas d’utilisation il peut être nécessaire de capter l’évènement pour utiliser ses propriétés.

Par exemple dans un évènement de scroll il y a des propriétés correspondant à la position du scroll, dans un évènement click nous pouvons retrouver les propriétés correspondantes aux coordonnées du click.

Dans l’exemple suivant nous mettons en place un bouton ainsi qu’un champ de texte dans le Template HTML, puis côté JS nous allons écrire des fonctions permettant de capter ces évènements.



Fichier HTML :

Le 1er v-on permettra de capter l’évènement click, le second capter l’évènement input (quand l’utilisateur tape au clavier dans le champ de texte)

<div id="monApp" class="container mt-4 mb-4 p-3 d-flex justify-content-center">

<div class="btn-group" role="group" aria-label="Basic mixed styles example">

<h3 class="mx-3">le nombre : {{leNombre}}</h3>

<button v-on:click="capterEventClick($event)" type="button" class="btn btn-info">test</button>

<input v-on:input="capterEventInput($event)" type="text">

<h3 class="mx-3">{{uneString}}</h3>

</div>

</div>

Fichier JS :

Vue.createApp({

data(){

return {

leNombre : 0,

uneString: 'Hello'

};

},

methods: {

capterEventClick(event){

console.log(event)

},

capterEventInput(event){

console.log(event);

this.uneString = event.target.value

}

}

}).mount('#monApp');

En mettant en place des fonctions dans lesquelles on capte les évènements observons les console log qui sont générés lorsque l’on click et l’lorsqu’on écrit dans l’input.

Dans la fonction capterInputEvent on récupère la value contenu dans l’evenement input pour l’assigner à la variable uneString.

Propriété de l’event Click (gauche) et Propriété de l’event input(droite)

