Memo VUEJS 2

Table des matières

[Hello world 2](#_Toc81930312)

[Directive 2](#_Toc81930313)

[Instance et cycle de vie 3](#_Toc81930314)

[Exemple avec création d’un timer 5](#_Toc81930315)

[Propriétés combinées : computed 5](#_Toc81930316)

[Propriété watcher : watch 5](#_Toc81930317)

Hello world : on créer une div#app, puis on vient initialiser notre vue

|  |  |
| --- | --- |
| **Ajout CDN** | <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js" defer></script> |
| **HTML** | <div id="app">          <a v-bind:href="link">{{ message }}</a> //on peut aussi écrire :href      </div>  *on ne peut pas interprété une variable dans un attribut avec {{ }} il faudra utiliser la directive v-bind* |
| **JS** | var vm = new Vue({      el: '#app', // selection du bloc      data: {        message: 'Hello Vue !', // notre variable  link: "http://google.fr"      }    }) |

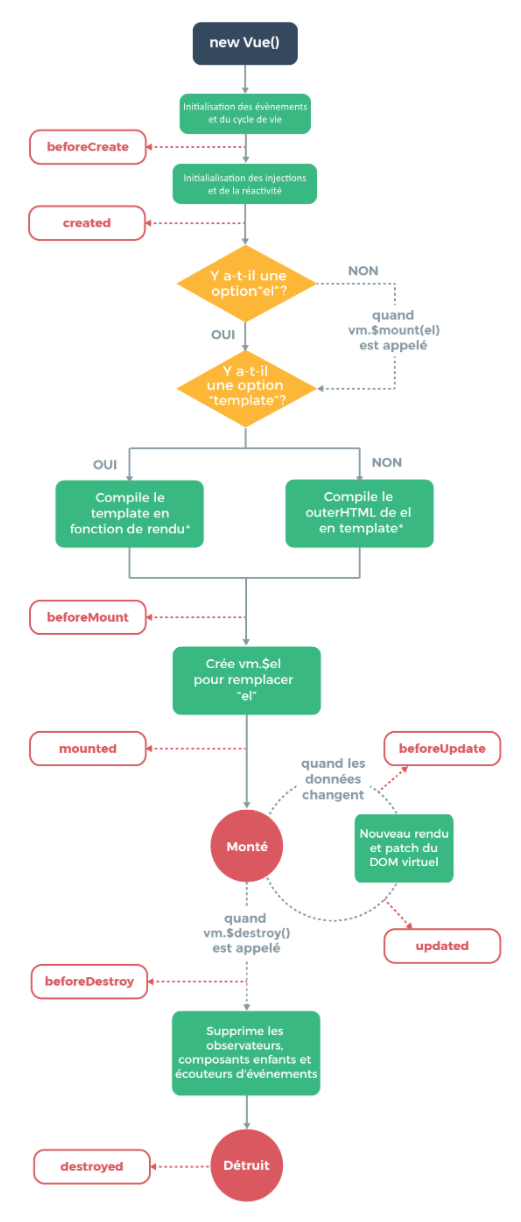
*On peut, dans la console appeler par exemple vm.$data, vm.$el pour accéder a des élément de vue*

# Directive

|  |  |
| --- | --- |
| **v-bind** : donné valeur attribut | <a v-bind:href="link">{{ message }}</a>  <a :href="link">{{ message }}</a> // propriété raccourcis  <p :class="{success: success, error: !success}"></p>  // vaudra success si success = true … |
| **v-if** : conditionnel  **v-else** : sinon | <p v-if="success">bravo</p> // s’affiche si success = true  <p v-else>echec</p> |
| **v-show** : idem mais l’element sera quand même dans le HTML (en dysplay none) | <p v-show="success">bravo</p> // s’affiche si success = true |
| **v-for** : boucle | <li v-for="person in persons">{{ person }}</li> |
| **v-on :click** : ajoute un évènement  Coté JS : on ajoute une propriété **methods** qui contiendra la méthode appelé lors du click. This permet d’accédé aux propriétés de **data** | <p v-on:click="close">clique moi</p>  <p @:click="close">clique moi</p> // racourcis |
| var app = new Vue({      el: '#app',      data: {        success: false,      },      methods:{        close: function(){          this.success = true        }      }    }) |
| **v-model** : lié une valeur d’un input à une variable de **data**  ici **cls** vaudra success si la case est coché sinon error au lieu de true/false | <input type="text" v-model="message">  <input type="checkbox" v-model="cls" :true-value="'success'" :false-value="'error'"> <span>{{cls}}</span> |

# Une image contenant texte Description générée automatiquementInstance et cycle de vie

Quand on créer une instance de Vue, il faut prendre en compte le cycle de vie du composant. On peut intervenir sur les étapes du processus. Par exemple on peut intervenir quand le composant est monté avec **mounted** pour recuperé le composant.



## Exemple avec création d’un timer

|  |  |
| --- | --- |
| Ici on créer un timer dont la valeur est stocké dans seconds.  Quand l’élément est prêt (**mounted**) on declenche un interval pour mettre à jours la valeur de seconds.  Et on arrête l’interval si l’instance est détruite (**destroyed**) | var vm = new Vue({      el: '#app',      data: {        seconds: 0      },      mounted: function(){        this.$interval = setInterval(() =>{          this.seconds++        },1000)      },      destroyed: function (){        clearInterval(this.$interval)      }    }) |

# Propriétés combinées : computed

|  |  |
| --- | --- |
| **Computed** : permet de déclencher l’exécution d’une méthode **seulement si** une des propriétés utilisées par la méthode est modifier (contrairement a **methods** qui se déclenchent **pour toute** modifications d’un attribut dans data) | computed:{      cls: function (){        console.log('cls called')        return this.success === true ? 'success' : 'error'      }    } |

Une image contenant texte

Description générée automatiquement

En revanche, avec en définissant la fonction comme ci dessus il n’est pas possible comme ça de modifier une valeur (set) mais seulement d’y accéder (get).

Pour pouvoir accéder au set et au get on pourra faire comme dans l’exemple ci-contre.

# Propriété watcher : watch

|  |  |
| --- | --- |
| **Watch** : permet de « surveiller » les modifications d’une propriété. Ici quand la valeur de la propriété success sera modifier on affichera sa valeur dans la console | watch: {        success : function(value){          console.log('watch', value)        }      } |