Cours 2 : directives

1. Problématiques

Il est actuellement impossible d'afficher un élément de la page selon une condition. Pour rappel, une condition est une structure de contrôle qui associe une suite d'instructions à une condition. Si la condition est réunie, alors les instructions sont exécutées.

Il est actuellement impossible de créer des boucles en HTML. Pour rappel, une boucle est une structure de contrôle qui exécute une suite d'instruction pour chaque élément d'un ensemble donné. C'est une structure très utilisée dans la plupart des applications web : affichage similaire de l'ensemble des publications sur un réseau social par exemple. Nous aurions alors besoin d'un moyen simple d'en créer, afin de rendre un ensemble d'éléments dans le DOM pour chaque élément d'un tableau.

Il est impossible d'associer la valeur d'une variable à un attribut HTML, rendant difficile la maintenance de nombreuses applications, particulièrement des applications graphiques.

2. Les directives VueJS

Comme vu dans le premier cours, VueJS met à notre disposition un outil très puissant pour cela : les directives. Ce sont des attributs HTML (pour rappel un attribut est composé d'un nom et d'une valeur) ayant des noms déterminés (souvent précédés du préfixe **v-**) et capables d'effectuer des actions définies soit par l'utilisateur (si vous créez vos propres directives, ce qui ne sera pas abordé à cette séance), soit par la librairie elle-même.

3. Associations de valeurs

La directive **v-bind** permet d'associer la valeur d'une variable VueJS à un attribut d'un élément du DOM. Cela peut se faire de plusieurs manières. Tout d'abord la syntaxe classique

```
<div v-bind:class="myVariable”></div>
```

Dans cet exemple, **myVariable** est une variable décrite dans le script associé à la balise **template** dans lequel on trouve cet élément. Il est alors possible de l'associer à l'attribut **class** de l'élément. Lorsque la variable sera mise à jour, la classe sera mise à jour. L'attribut sera transformé de **v-bind:class** en **class** dans le DOM, et l'application VueJS se chargera de maintenir l'attribut à jour d'après la valeur de la variable.

Il existe une forme raccourcie de cette syntaxe, où vous omettez le **v-bind** :

```
<div :class="myVariable"></div>
```

Remarquez la présence des deux points devant le nom de l'attribut, cela signifie que la valeur passée sera évaluée dans le contexte du composant, en tant qu'expression JavaScript.

Il existe enfin une dernière façon d'utiliser v-bind, en lui passant directement un objet JavaScript (équivalent à un tableau associatif en PHP, ou à une Map en Java) qui représentera les couples

attributs/valeurs, les clefs de l'objet étant les attributs, les valeurs étant les valeurs associées. Ces valeurs seront alors lues comme des expressions JavaScript, par exemple, la syntaxe suivante est équivalent aux deux premières :

```
<div v-bind="{ class : myVariable }”></div>
```

Notez que si vous vouliez passer une chaîne de caractère en dur à v-bind, il serait nécessaire de la placer entre simple quotes, si la valeur entière de **v-bind** est comprise entre double quotes.

4. Le rendu conditionnel

3.1. v-if

La directive permettant le rendu conditionnel s'appelle **v-if**. Elle fonctionne exactement comme une condition dans un langage de programmation classique. On lui donne comme valeur une expression JavaScript qui sera évaluée dans le contexte courant de l'application VueJS correspondante. Par exemple, pour afficher un bouton, vous pouvez utiliser la syntaxe suivante :

```
<button v-if="condition">Bouton</button>
```

Dans cet exemple, **condition** est un booléen, s'il a pour valeur **true**, alors le bouton sera inséré dans le DOM, sinon il ne sera pas inséré du tout. Notez qu'il ne s'agit pas de l'afficher ou de le masquer avec du CSS (ce que nous verrons plus loin), mais réellement de ne pas créer le nœud dans le DOM.

Vous pouvez appeler une fonction, une variable, ou une **ref** dans un **v-if** comme dans la plupart des expressions.

3.2. v-else

De la même façon que **v-if** est analogue à une condition, **v-else** est analogue à une alternative. Il doit être placé sur l'élément HTML directement situé après l'élément portant le v-if pour être correctement interprété. Il ne peut pas être utilisé sans un **v-if** préalable. Si vous voulez afficher un élément ou un autre selon une condition, la syntaxe sera donc :

```
<div v-if="condition">Si condition vraie</div>
<span v-else>Si condition fausse</span>
```

3.3. v-else-if

Quand plusieurs branches conditionnelles sont dépendantes les unes des autres, vous pouvez utiliser v-else-if pour chaque branche conditionnelle supplémentaire. La syntaxe **v-if**, **v-else-if**, **v-else** correspond à la syntaxe **switch/case** dans la plupart des langages usuels de programmation.

5. Les boucles

Pour représenter les éléments d'une liste, VueJS nous offre la directive **v-for**. Cette directive permet d'afficher un élément dans le DOM pour chaque élément de la liste itérée. Chaque élément HTML ainsi inséré doit également posséder un attribut **:key** unique, utilisé par VueJS pour ne pas avoir à redessiner toute la liste lors de sa modification. Deux syntaxes sont disponibles. Si la variable **myVariable** contient un tableau de chaînes de caractères à itérer, alors **v-for** peut être utilisée de deux façons différentes :

```
<div v-for="item in myVariable :key=”item”>{{ item }}</div>
<div v-for="item of myVariable :key=”item”>{{ >{{ item }}</div>
```

Il est également possible d’exploiter les indices du tableau avec la syntaxe suivante :

```
<div v-for="(item, index) in myVariable” :key=”index”>{{ item }}</div>
```

6. Combinaisons de boucles et conditions

Il est important de noter qu’il est **fortement** déconseillé d’appliquer **v-if** et **v-for** sur le même conteneur, cela provoquant la plupart du temps des erreurs inattendues pour des raisons de priorité des directives. Si vous devez le faire, il vous est conseillé d’utiliser une balise **template** la balise **template** sera ensuite supprimée lors du rendu. Par exemple, pour ne rendre dans le tableau suivant que les éléments supérieurs à 10 :

```
const arr = [ 3, 5, 15, 10, 9 ] ;
```

On pourra utiliser la syntaxe :

```
<template v-for="(item, index) in arr” :key=”index”>
  <div v-if=”item > 10”>{{ item }}</div>
</template>
```