Vue JS

Les Composants avancés

Précédemment, nous avons aperçu comment fonctionnent basiquement les composants Nous avons principalement vu tout ce qui concerne les templates Il est maintenant temps d'évoluer et d'utiliser toutes la puissance des composants dans un projet Vuejs.

Lors de la création du projet via la CLI, un composant a été créé; Le composant `HelloWorld`.

Dans Vuejs, comme d'autres frameworks, les composants ont pour vocation d'être réutilisés. Le but est d'avoir un composant pour une opération.

Dans certains cas, on peut parler d'Atomic Design. Le principe est simple Un composant gère la logique, le design et le rendu d'un élément Ce composant est appelé à chaque fois que l'on doit afficher cet élément Par exemple:

- un composant affiche une liste
- Chaque item de la liste est un nouveau composant

Commençons par créer et faire fonctionner un premier composant. Un composant fonctionnel est composé de trois éléments:

- Le HTMl dans la balise template
- La logique JavaScript dans la balise script
- Le design CSS dans la balise style

Dans components créer un fichier `Home.vue` :

```
// src : components/Home.vue
<template>
  <div class="hello">
    <h1>Hello World</h1>
  </div>
</template>
<script>
  export default {
    name: 'Home',
  }
</script>
```

Avec ce code, nous venons de créer un composant simple, sans logique, et affichant un titre Hello World. Même si ce composant est créé, il n'est pas encore actif. Pour cela, il faut l'importer dans le fichier parent puis l'instancier et enfin le définir dans le HTML

Allez dans le fichier `App.vue`. Dans la balise script, avant le export ajoutez le code suivant

```
// src : App.vue
import Home from './components/Home'
```

Dans le export ajoutez :

```
// src : App.vue
  components: {
    Home,
  },
```

Et enfin, dans le HTML, affichez le composant

```
// src : App.vue
<home></home>
```

Exercice: 20min

- Ajoutez un composant sur le projet. Pour le moment, ce composant ne va afficher qu'un titre
- Ajouter d'autres composants ou réutilisez le même plusieurs fois.

Passage d'attributs à un composant

Pour le moment, notre composant n'effectue rien de plus qu'afficher du HTML. Vous pouvez aussi ajouter des fonctionnalités et du CSS à ce composant. Mais bien souvent, il peut être utile de transférer des informations depuis un composant parent à un composant enfant. Pour cela, nous utilisons le principe de `props`.

- Il faut voir cette `props` comme une propriété ayant un nom et une valeur. Au niveau du parent, cela est défini simplement dans le HTML `<home msg="mon message"></home>`

Afin de récupérer le contenu passé du parent à l'enfant, il est nécessaire d'utiliser le mot-clef props dans le export de l'enfant. Il existe plusieurs façon de définir et récupérer les props:

- Par un tableau
- Par un objet

Nous allons préférer passer par l'objet, car cela permet de gérer la validation des types.

Exercice: 40min

Utilisez l'API en GET pour récupérer des lieux de tournage de films et les lister via différents composants Vuejs.

Nous n'allons pas directement interroger une API pour le moment mais uniquement lire un fichier disponible dans resources/

Vous allez devoir réaliser :

- Un composant pour la liste
- Un composant par item

** Exemple : exercices/composants-avances

Filtres

Des fois, les données que vous récupérez sont bien, mais vous voulez les afficher différemment : Par exemple pour une data ou pour une somme d'argent avec le placement du symbole ($100 ou 100€)

Le role des filters est de permettre d'apporter des modifications simples sur les variables que vous affichez dans le HTML..

L'utilisation se fait en deux étapes :

- Créer le filtre dans le export

```
filters: {
  currency: (value) => {
    return `${value}€`
  },
}
```

- Utiliser le filtre dans le code HTML

```
<p>{{ price | currency }}</p>
```

- Les filtres sont des fonctions JavaScript simple :
  - Ils ont le même comportement
  - Dans le HTML, vous pouvez passer des params au filter `price | currency(param)`

Exercice: 30min

Sur le code précédent essayez de réaliser un filtre sur la data issue du json

- Affichez le date de début du tournage au sein de votre component et appliquez un filtre affichant la date au format français jj/mm/yyyy
- Ajoutez des paramètres aux filtres pour pouvoir spécifier le format attendu. Le vanilla JS ne permets pas de manipuler des dates aisément. La librairie `moment` peut vous aider et rajouter des fonctionnalités bien pratiques pour celà.

Exemple: exercices/composants-avances

Cycle de vie

- Chaque instance de vue traverse une série d'étapes d'initialisation au moment de sa création
- Elle doit mettre en place l'observation des données, compiler le template, monter l'instance sur le DOM et mettre à jour le DOM quand les données changent
- En cours de route, elle va aussi invoquer des hooks de cycle de vie, qui nous donnent l'opportunité d'exécuter une logique personnalisée à chaque niveau.

Il existe 4 hooks principaux :

- created
- mounted
- updated
- destroyed

Souvent, le cycle de vie `mounted` est utilisé pour invoquer les méthodes permettant de récupérer des données provenant d'une API. Le cycle de vie `updated` sert lors du rechargement d'un composant Le cycle de vie `destroyed` permet d'appeler une méthode lorsque l'on change de composant

Exercice: 20min

- Testez les cycles de vie. Peut-être plus aisé sur l'exercice des courses.
- Modifiez les composants pour que l'appel à l'API s'effectue tout seul au chargement

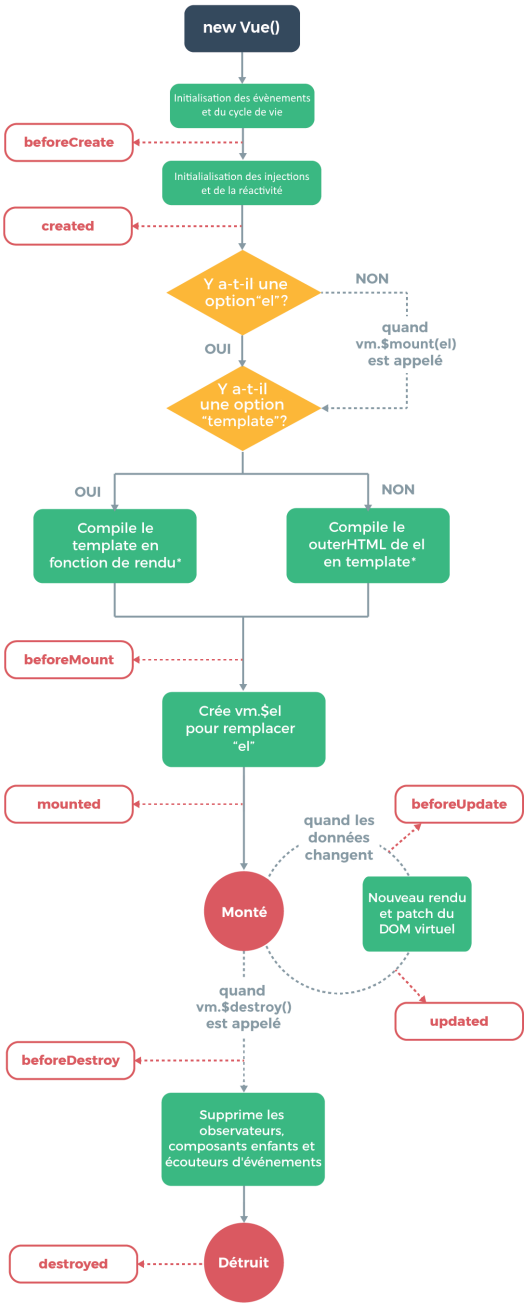

Figure 1: VueJS Lifecyle