Les composant vue.js

Vous connaissez sans doute les **Web Component,** nouveau standard qui permet d'enrichir le HTML de façon modulaire. En gros on peut créer une entité qui utilise des éléments HTML et des fonctionnalités propres, le tout étant facile à intégrer avec une simple balise personnalisée.

On va voir dans ce chapitre que Vue.js propose une approche simple et efficace très fortement inspirée de **polymer**.

Un composant est un élément isolé, autonome, qui possède des propriétés et des comportements et qu'on peut utiliser directement dans le HTML…

Un premier composant

```
Vue.component('mon-composant', {
  template: '<p>Mon premier composant !</p>'
});
```

Il faut aussi initialiser la VueModèle comme d'habitude :

```
new Vue({
  el: '#tuto'
});
```

Ensuite on utilise la balise personnalisée dans le HTML :

```
<div id="tuto">
  <mon-composant></mon-composant>
</div>
```

Passage de données

Il arrive souvent qu'on ait besoin de transmettre des informations pour renseigner le composant. On va ajouter une propriété au composant :

```
Vue.component('nom', {
  props: ['nom'],
  template: '<p>Mon nom est {{nom}}</p>'
});
```

```
<div id="tuto">
  <nom nom="Toto"></nom>
</div>
```

Attribut dynamique

```
<div id="tuto">
  <input v-model="nomSaisi">
```

```
 <br>
 <nom :nom="nomSaisi"></nom>
</div>
```

```
Vue.component('nom', {
 props: ['nom'],
 template: '<p>Mon nom est <strong>{{nom}}</strong></p>'
});

new Vue({
 el: '#tuto',
 data: { nomSaisi: '' }
});
```

gestion d'une liste

```
Vue.component('liste', {
 props: ['personne'],
 template: '<li>{{personne.nom}} {{personne.prenom}}</li>'
});

new Vue({
 el: '#tuto',
 data: {
  personnes: [
   {nom: "Durand", prenom: "Jacques"},
   {nom: "Dupont", prenom: "Albert"},
   {nom: "Martin", prenom: "Denis"},
  ]
 }
});
```

```
<div id="tuto">
 <liste v-for="personne in personnes" :personne="personne"></liste>
</div>
```

Exercice : changer l'affichage sous forme de tableau html

Vous voyez que la création du template à l'intérieur du composant est problématique.
Il est possible d'externaliser ce Template : On peut inclure le *template* dans le HTML avec la nouvelle balise **template** qui est maintenant bien prise en charge par les navigateurs.

Exercice :Modifier votre exercice précédent en externalisant le template.

```
<template id="t_template"> ….</template>
```

Exercice de synthèse : interrogation de l'API google book

1. Testez l'API de google Book avec
   https://www.googleapis.com/books/v1/volumes?q=inauthor:musso
   Le retour est au format Json, vous pouvez découvrir le reste de l'API :
   https://developers.google.com/books/docs/v1/using

2. Faites des recherches sur ajax et vue.js et sur le parsing json

3. L'objectif est de mettre en page les résultats concernant un auteur passer en paramètre par le biais d'un input dans la même page.

On utilisera obligatoirement les components , associé à des templates.

Les éléments à afficher sont :

- Le titre
- L'année de parution
- Un descriptif (limité à 100 caractère) => on utilisera un **custom filter** vue.js
- L'image de la couverture
- Un click sur le titre amènera sur la page d'achat google du livre.
- Toutes les autres propositions seront étudiés ☺