TP3 VueJS

- **framework évolutif** pour construire des interfaces utilisateur
- bibliothèque concentrée sur la partie vue
- Découpage en composants
- VueJs : Approche simple. Application existante pour gérer la vue. Modulable et simple d'utilisation.

Initialisation VueJS:

- Importer VueJs
- Créer un élément avec un id particulier
- El : sur quel élément greffer vueJS
- Data : tableau qui a toutes les variables
-

```
var vm = new Vue({
  // options
})
```

Premier exemple :

```
<div id="app">
  {{ message }}
</div>
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue !'
  }
})
```

Ajouter vue avec :

```
<script src="https://cdn.jsdelivr.net/npm/vue"></script>
```

Directives :

Les directives permettent de greffer un comportement spécial.

v-bind : lier des éléments
v-if : condition
v-for : afficher une liste d'éléments en provenance d'un tableau de données

question :
1. quelle est la différence entre v-if et v-show ?
2. Faire un <ul> avec une liste de personnes

Entrées utilisateurs :

- v-on : pour attacher des écouteurs d'évènements qui invoquent des méthodes sur nos instances de Vue
- v-model : fait de la liaison de données bidirectionnelle entre les champs d'un formulaire et l'état de l'application

Les composants

les composants sont des éléments personnalisables auxquels le compilateur de Vue attache un comportement.
Un composant permet de créer des nouvelles balises HTML avec un comportement spécifique personnalisé.
Tous les composants Vue sont également des instances de Vue
Le composant doit inscrit avant l'instanciation de l'instance racine de Vue.

Pour inscrire un composant global : `Vue.component(tagName, options)`

```
Vue.component('my-component', {
  // options
})
```

un composant peut être utilisé dans le template d'une instance en tant qu'élément personnalisé, `<my-component></my-component>`

A tester :

```
<div id="example">
  <my-component></my-component>
</div>
```

```
// inscrire
Vue.component('my-component', {
  template: '<div>Hello World!</div>'
})

// créer une instance racine
new Vue({
  el: '#example'
})
```

Props

Une prop est un attribut personnalisé permettant de passer des informations depuis des composants parents.

```
Vue.component('child', {
  props: ['message'],
  template: '<span>{{ message }}</span>'
})
```

```
<child message="bonjour !"></child>
```

Les formulaires :

v-model ne prend pas en compte la valeur initiale des attributs value, checkedou selected fournis par un champ.

- **texte**

```
<input v-model="message">
<p>Le message est : {{ message }}</p>
```

- **radio**

```
<div id='example-3'>
  <input type="checkbox" id="BD" value="BD" v-model="checkedNames">
  <label for="BD">BD</label>
  <input type="checkbox" id="systeme" value="systeme" v-model="checkedNames">
```

```
  <label for=" systeme "> systeme </label>
  <input type="checkbox" id="reseaux" value="reseaux" v-model="checkedNames">
  <label for=" reseaux "> reseaux </label>
  <br>
  <span>Noms cochés : {{ checkedNames }}</span>
<div>
new Vue({
  el: '#example-3',
  data: {
    checkedNames: []
  }
})
```

Exrecice 1 :

Réaliser un formulaire avec deux zones de saisies login et mot de passe. Le mot de passe est affiché lors d'un clic sur un bouton.

Exrecice 2 :

Réaliser une todo list