TP 2
EFREI 3A WEB

Louis Cherel - 08/2020

Préambule

Merci de lire **toutes les consignes**, elles sont écrites pour être lues.

Cf les consignes du TP 1 pour le but et le séquencement des TP.

Les ressources conseillées

Vos armes seront donc le site officiel de Vue.js, Mozilla Developer Network (MDN), Stack Overflow, et les recherches en anglais. **Évitez au maximum** les forums (pas les cours) des sites comme OpenClassrooms ou Comment Ça Marche, qui proposent des solutions souvent peu fiables. W3Schools peut être une bonne ressource, mais les bonnes pratiques et les standards ne sont pas leur fort. Leur préférer le MDN lorsque possible.

Quantité de blogs sont très bons, notamment alsacreations (en français) ou css-tricks (en anglais).

Exercices

Exercice 1 (🕒 30 min lecture + 🕒 15 min pratique)

Lectures préalables nécessaires:

- Vue.js introduction,
- Vue.js Instance de vue,
- Vue.js Syntaxe de template

1) **Il est vraiment primordial** que vous lisiez les trois lectures préalables, sans quoi ces exercices n'auront pas beaucoup de sens
2) Créez un fichier .html avec votre éditeur préféré, et ajoutez la structure de base d'un fichier (html, head, body)
3) Ajoutez Vue.js dans le <head> du fichier:

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

4) Ajoutez une balise <script> dans le <body>

5) Notre objectif est d'afficher un âge dépendant d'une variable dans notre page

    a) En Javascript, créez une instance de Vue, qui sera stockée dans la variable **vm**, avec dans l'objet *data* une variable "age" dont la valeur correspond à votre âge

        i) Si vous ne comprenez pas les termes employés, relisez bien la page "Instance de vue" du guide de Vue.js

    b) Dans le HTML, créez une balise qui contiendra notre template, et indiquez "Mon âge est {{ age }}"

        i) N'oubliez pas d'indiquer un id sur votre balise , et de référencer cet id dans l'instance de vue avec la propriété *el*

6) Nous allons ajouter un bout de code permettant de changer la valeur de notre âge toutes les secondes

    a) Créez une fonction oneMore() dont le but est d'augmenter la variable vm.age de 1

    b) avec setInterval, déclenchez votre fonction oneMore toutes les secondes

7) Que se passe-t-il à l'écran lorsque vous lancez le script dans votre navigateur ?

Exercice 2 (🕒 30 min lecture + 🕒 20 min pratique)

Lectures préalables nécessaires:

- Vue.js: Propriétés calculées et observateurs,
- Vue.js: Liaisons de classes et de styles
- Vue.js: Rendu conditionnel

1) Créez un nouveau fichier .html pour cet exercice, en faisant bien attention à inclure les balises minimales et le <script> de vue.js

2) Ajoutez l'instance de vue dans une nouvelle balise <script>

a) Votre instance de vue doit contenir un attribut `isWhite: false` dans l'objet *data*

3) dans votre instance de vue, ajoutez un attribut "methods", et ajoutez-y une méthode inverser() comme ceci:

```
methods: {
  inverser() {
    this.isWhite = !this.isWhite
  }
}
```

4) Vous pouvez maintenant attacher cette méthode à un bouton, afin de laisser l'utilisateur agir sur cette propriété:

```
<button v-on:click="inverser">Changer la couleur</button>
```

a) Lorsque l'utilisateur cliquera sur le bouton, la méthode `inverser` sera appelée

5) Faites en sorte, avec les directives v-if et v-else, d'avoir un texte qui affiche soit "Blanc" soit "Noir" en fonction de la propriété `isWhite`.

6) grâce à v-bind:style ou v-bind:class, faites en sorte que le texte soit blanc sur fond noir ou noir sur fond blanc en fonction de `isWhite`.

7) Créez une propriété calculée myText qui retourne la chaîne "Blanc" si isWhite est à true, "Noir" sinon. Affichez cette propriété calculée dans le template

Exercice 3 (🕒 20 min lecture + 🕒 20 min pratique)

Lectures préalables nécessaires:

- Vue.js: Rendu de liste
- Vue.js: Gestion des évènements

1) Créez un nouveau fichier .html pour cet exercice, en faisant bien attention à inclure les balises minimales et le <script> de vue.js

2) Créez une instance de vue, et ajoutez une propriété ***fruits*** à *data*, qui sera un tableau contenant 5 chaînes de caractères nommant des fruits: ['Pomme', 'Orange', 'Fraise', 'Melon', 'Cerise']

3) Créez dans votre template une liste HTML <ul> qui contienne l'intégralité des fruits de votre liste ***fruits*** dans des <li>, grâce à la directive *v-for*

4) Ajoutez un attribut `currentFruit: 'Pomme'` à *data,* et affichez sa valeur dans le template

Pomme

- Pomme
- Orange
- Fraise
- Melon
- Cerise

5) Concevez une méthode `changeCurrentFruit(fruit)`, comme vu dans l'exercice 2, qui reçoive un argument *fruit*, et qui change la propriété `currentFruit` pour lui affecter le fruit reçu en paramètre

6) Faites en sorte que lorsqu'on clique sur l'un des fruits de la liste HTML, la méthode `changeCurrentFruit` soit appelée avec comme paramètre le fruit actuel dans la liste

    a) Dit autrement, lorsqu'on clique sur le fruit "orange", changeCurrentFruit reçoit "orange" comme paramètre, et currentFruit est donc réaffectée à "orange"

7) Trouvez un moyen pour que l'élément de la liste HTML qui correspond au fruit de `currentFruit` soit affiché en gras, et seulement lui.

    a) Lorsqu'on clique sur un autre fruit, cet autre fruit doit être mis en gras et le fruit précédent doit retrouver sa forme normale

Pomme

- **Pomme**
- Orange
- Fraise
- Melon
- Cerise

Fraise

- Pomme
- Orange
- **Fraise**
- Melon
- Cerise

8) Ajoutez une croix ❌ à gauche de chaque fruit dans le template, et faites en sorte que lorsque l'utilisateur clique dessus, l'élément sous le curseur

soit supprimé. Attention à bien supprimer une string du tableau fruits et non pas un élément HTML.

Exercice 4 (🕒 20 min lecture + 🕒 20 min pratique)

Lectures préalables nécessaires:

- Vue.js: Liaisons sur les champs de formulaire

Cet exercice sera un peu moins guidé que les autres, il va falloir trouver par vous-mêmes les éléments à configurer pour obtenir le résultat escompté

1) Créez un nouveau fichier .html pour cet exercice, en faisant bien attention à inclure les balises minimales et le <script> de vue.js

2) Ajoutez l'instance de vue, et dans l'objet data, ajoutez un attribut ***cours*** qui contient les éléments suivants:

```
['Algorithmique', 'Structures de données', 'Programmation Web', 'Web Avancé', 'Frond-end Web Development', 'Asynchronous Event-Driven Programming with Node.js']
```

3) Comme dans l'exercice précédent, faites en sorte d'afficher ces cours, mais pas dans une liste <ul> cette fois-ci, en temps que paragraphes dans un <article>, toujours avec v-for

4) Grâce à ce que vous avez appris sur les champs de formulaires, avec *v-model*, ajoutez un champ de texte et un bouton permettant d'ajouter un élément dans la liste de cours

5) Ajoutez un menu déroulant de type <select>, qui liste tous les cours dans notre liste Javascript, un nouveau champ de texte, ainsi qu'un nouveau bouton "Modifier"

6) Il faut maintenant que lorsque l'utilisateur sélectionne l'un des cours avec la liste déroulante, il puisse modifier le texte de l'élément de la liste avec le nouveau champ de texte et valider avec le bouton "Modifier"

    a) Si vous faites ça correctement, tous les endroits où la liste est rendue devraient se mettre à jour avec le nouveau texte

Exercice 5 (🕒 30 min pratique)

L'objectif de cet exercice est de réaliser une ToDo list avec Vue.js.

Voici le cahier des charges

- Un élément de ToDo list contient: un titre, un contenu, et un booléen indiquant si la tâche a été effectuée
- Votre application doit proposer un formulaire permettant d'ajouter un item à votre todo list
- Votre application doit lister tous les items de votre ToDo list, avec une coche si la tâche a été réalisée
- L'utilisateur doit pouvoir changer l'état d'un item de la liste de "non effectué" à "effectué

À vous de jouer.