PW 2 EN
EFREI 3A WEB

Louis Cherel - 08/2020

Préambule

Please read all the instructions, they are written to be read.

Cf the instructions in PW 1 for the purpose and sequencing of the PWs.

Les ressources conseillées

Your weapons will be the official site of Vue.js, Mozilla Developer Network (MDN), Stack Overflow, and English searches. **Avoid as much as possible** the forums (not the courses) of sites like OpenClassrooms or How It Works, which offer often unreliable solutions. W3Schools can be a good resource, but best practices and standards are not their forte. They prefer DND when possible.

Many blogs are very good, including alsacreations (in French) or css-tricks (in English).

Exercises

Exercise 1 (🕒 30 min read + 🕒 15 min practice)

Necessary preliminary reading:

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

1) **It's really important** that you read the three readings beforehand, otherwise these exercises won't make much sense

2) Create an .html file with your favorite editor, and add the basic structure of a file (html, head, body)

3) Add Vue.js in the <head> of the file:

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

4) Add a <script> tag in the <body>

5) Our goal is to display a variable-dependent age on our page

    a) In Javascript, create an instance of View, which will be stored in the vm variable, with in the data object an "age" variable whose value corresponds to your age

        i) If you don't understand the terms used, re-read the "View Instance" page in the Vue.js guide

    b) In the HTML, create a tag that will contain our template, and indicate "My age is {{ age }}"

        i) Don't forget to indicate an id on your tag, and to reference this id in the view instance with the property el

6) We are going to add a piece of code allowing us to change the value of our age every second

    a) Create a oneMore() function whose goal is to increase the variable vm.age by 1

    b) with setInterval, trigger your oneMore function every second

7) What happens on the screen when you run the script in your browser?

Exercise 2 (🕒 30 min read + 🕒 20 min practice)

Necessary preliminary reading:

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

1) Create a new .html file for this exercise, being careful to include the minimal tags and the <script> of Vue.js

2) Add the view instance in a new <script> tag

    a) Your view instance must contain an `isWhite: false` attribute in the *data* object

3) in your view instance, add a "methods" attribute, and add an invert() method like this:

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

4) You can now attach this method to a button, in order to let the user act on this property:

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

   a) When the user clicks on the button, the method `inverser` will be called

5) With the v-if and v-else directives, make sure you have a text that displays either "White" or "Black" according to the property `isWhite`.

6) With v-bind:style or v-bind:class, make the text white on a black background or black on a white background according to `isWhite`.

7) Create a myText calculated property that returns the string "White" if isWhite is true, "Black" otherwise. Display this calculated property in the template

Exercise 3 (🕒 20 min read + 🕒 20 min practice)

Necessary preliminary reading:

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

1) Create a new .html file for this exercise, being careful to include the minimal tags and the <script> of Vue.js

2) Create a view instance, and add a **fruit** property to *data*, which will be an array containing 5 strings naming fruits: ['Apple', 'Orange', 'Strawberry', 'Melon', 'Cherry']

3) Create in your template an HTML list <ul> that contains all the fruits of your ***fruit*** list in <li>, thanks to the *v-for* directive

4) Add an attribute `currentFruit: 'Pomme'` to *data,* and print its value in the template

Pomme

- Pomme
- Orange
- Fraise
- Melon
- Cerise

5) Conceive a method `changeCurrentFruit(fruit)`, as seen in Exercise 2, which receives a **fruit** argument, and which changes the property `currentFruit` to assign it the fruit received in parameter

6) Make sure that when one of the fruits in the HTML list is clicked on, the method `changeCurrentFruit` is called with the current fruit in the list as a parameter

    a) In other words, when you click on the fruit "orange", changeCurrentFruit receives "orange" as a parameter, and currentFruit is therefore reassigned to "orange"

7) Find a way to make the element in the HTML list that matches the `currentFruit` fruit appear in bold, and only it

    a) When you click on another fruit, this other fruit should be highlighted in bold and the previous fruit should return to its normal shape

Pomme

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

Fraise

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

8) Add a cross ❌ to the left of each fruit in the template, and make sure that when the user clicks on it, the element under the cursor is removed. Be careful to remove a string from the fruit table and not an HTML element

Exercise 4 (🕒 20 min read + 🕒 20 min practice)

Necessary preliminary reading:

- Vue.js: Liaisons sur les champs de formulaire

This exercise will be a little less guided than the others, you will have to find by yourselves the elements to be configured to obtain the expected result

1) Create a new .html file for this exercise, being careful to include the minimal tags and the <script> of Vue.js
2) Add the view instance, and in the data object, add a course attribute that contains the following elements:

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

3) As in the previous exercise, make sure to display these courses, but not in a list <ul> this time, as paragraphs in an <article>, always with v-for
4) Thanks to what you learned about form fields, with *v-model* add a text field and a button to add an item in the course list
5) Add a drop-down menu of type <select>, which lists all the courses in our Javascript list, a new text field, and a new "Edit" button
6) It is now necessary that when the user selects one of the courses with the drop-down list, he can modify the text of the list item with the new text field and validate with the "Modify" button
   a) If you do this correctly, all the places where the list is rendered should update with the new text

Exercise 5 (🕒 30 min practice)

The objective of this exercise is to create a ToDo list with Vue.js.
Here are the specifications

- A ToDo list item contains: a title, a content, and a boolean indicating if the task has been completed

- Your application must propose a form to add an item to your todo list
- Your application must list all the items in your ToDo list, with a check mark if the task has been completed
- The user must be able to change the status of an item in the list from "not done" to "done"