Unit 5

Classwork: The classwork for this unit should be saved in a new folder: **your_repo/unit_5/**
Homework: Homework should be saved here: **your_repo/homework/unit_5/**

*** Day 01 ***

Vue.js

We will take a break from Django, so we can learn Vue.js, but we will use Django later to serve the requests done by Vue.js and have a full stack application.

I saw the majority of students did not implement a good amount of JS components in your final project for the first semester. I understand you, maybe JS is not easy. Or maybe you did not have enough time.

I have good news about the frontend development. We are going to use a JS framework called Vue.js. You remember a **framework is a library that you can use to create an application**: like Django is a web framework built in Python.

We already learned some JS basics. This will be useful for Vue.js, as we have to write JS code. But Vue.js is going to help us to have a structure and take care of a lot of the JS in terms of refreshing some sections of the html by itself when data is going to change. This will be very helpful to create a more complicated and interactive frontend.

You can use Vue.js in some sections of your html page, for example in the menu bar, or you can use it for a whole page of your application. You can also have Vue.js in one or multiple pages. Or you can have your entire application done with Vue.js and this moment it will be called a single page application (SPA).

Also, there are 2 ways to use Vue.js in your application:

1. The first one, which is the easiest way, but limited in its use (especially in using some Vue.js libraries). It imports Vue.js into our HTML, so the developer writes the Vue code along with HTML.
2. The second way involves building the application that requires setting up a configuration for our Vue application, and especially being able to make it work with our Django application.

We are going to learn both ways.

First, let's go for the easiest configuration, where we do not need to build a Vue.js application.

Let's create the files: index.html and our_app.js

In our html, we will have a div with the id="app": `<div id="app">{{ message }}</div>`

index.html:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Movie Theater</title>
</head>

<body>
    <div id="app">{{ message }}</div>
    <script type="module" src="our_app.js"></script>
</body>

</html>
```

{{}} have more or less the same behavior than in Django.

As you can see the html code calls the script our_app.js which imports 2 functions from Vue.js: create the Vue.js app, and mount it on the div id="app".

our_app.js:

```
import { createApp, ref } from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

 createApp({
   setup() {
     const message = ref('Hello Vue!')
     return {
       message
     }
   }
 }).mount('#app')
```

Save your classwork here: **your_repo/unit_5/movie_theater/your_files_here**

Let's use the createApp differently now.

We are going to insert some values inside a list, using an input field, and click on a button to insert this value inside the list, and display the list in a tag:

```
<div id="app">
    <div>
      <input type="text" id="my_value" v-model="inputValue" />
      <button v-on:click="addValue">Add value</button>
    </div>
    <br/>
    <ul>
      <li v-for="value in myList">{{ value }}</li>
    </ul>
</div>
```

And our Javascript code:

```
import { createApp, ref } from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

createApp({
 data() {
   return {
     myList: [],
     inputValue: ""
   };
 },
 methods: {
   addValue() {
     this.myList.push(this.inputValue);
     this.inputValue = "";
   },
 }

}).mount('#app')
```

In our Vue app, we are going to introduce a data section and a methods section. We are using the structure of the Vue js framework.

In the data section, which is a function, we are going to return the initial state of our data, and define the name of our variables.

In the methods section, we define the function that our application will use.

Notice, I'm initializing a string and a list in the data section.

And in our html, I'm setting up an input field:
<input type="text" id="my_value" v-model="inputValue" />

We need to connect our html element with our vue js app, and for that, Vue.js introduces a lot of keys that begin usually with **v-something**, and there will be a shortcut with **@ or :** (we will see that later).

Here with an input field, you can connect the input with a Vue.js data variable with the key **v-model**. If you change the value in the html side, the value in javascript will be updated automatically. If you change the value in the javascript side, you will see it display automatically on the html. This is the magic of Vue js, it is going to take care of a lot of the implementation for us, and coding in Vue js will help us to let our creativity flow, while it takes care of the annoying stuff.

In our Vue app, we have defined our method addValue. It is going to take the value from the input field, and then add it to the list, and then we reinitialize the inputValue.
We connect this method to the html button with the key **v-on:click** (which means, when we click on this button, use this vue js method).

And then, we are going to display the list. We create a <ul> element, and we are going to create a vue js loop inside using <li> elements, with the key **v-for.** We are looping over the list myList using for each iteration the value 'value', and we are going to insert this value inside the <li> elements with {{ }}.

What do you think about this simple example?

Test the code on your side, check the html when you add a value on the list.

Homework (It is already posted on the website):

Research: How to retrieve the variables values myList and inputValue in the console (developer tool)? Once you succeed, change the inputValue, and push a value to myList. What does it happen when you do that? Is the html automatically updated?

Write your answers here: **your_repo/homework/unit_5/02_08_variables_console.txt (md or pdf file)**