

- [Question and respond on Lab3](#)
  - [Vue project setup](#)
    - [Question 1](#)
    - [Exercise 1](#)
    - [Question 2](#)
    - [Question 3](#)
    - [Question 4](#)
    - [Exercise 2](#)
    - [Exercise 3](#)
    - [Exercise 4](#)
  - [Base layout](#)
    - [Exercise 5](#)
    - [Questions 5](#)
    - [Exercise 6](#)
  - [Question 6](#)
  - [Exercise 7](#)
  - [Exercise 8](#)
  - [Exercise 9](#)
  - [Exercise 10](#)

## Question and respond on Lab3

---

- [Lab link](#)

## Vue project setup

---

### Question 1

That is the main difference between local installation and global installation of packages with npm? What kind of packages do you generally install locally? What kind is generally installed globally?

The main difference between local installation and global installation of packages with npm is that the local installation is only available for the project where it is installed.

The global installation is available for all projects on the computer. The same parameters are used for both installations.

## Exercise 1

Create a new Vue project (called vue-oauth-microsoft-graph). Opt for the Vue3 recipe that relies on webpack and babel for the build chain.

```
npm install -g @vue/cli
# verification que vue est installée
vue --version
# creation du projet
vue create vue-oauth-microsoft-graph
```

## Question 2

Webpack is internally used by the Vue CLI. Why is it required to deal with both multiple JavaScript files and special extensions like .vue?

Webpack is required to deal with both multiple JavaScript files and special extensions like .vue because it allows to bundle all the files into one file. It also allows to use the latest JavaScript features and to use a modular approach.

## Question 3

What is the role of babel and how browserslist may configure its output?

Babel is a JavaScript compiler. It allows to use the latest JavaScript features and to use a modular approach. Browserslist is a configuration file that allows to specify the browsers that the project must support. Babel will then compile the code to be compatible with the specified browsers.

## Question 4

What is eslint and which set of rules are currently applied? The eslint configuration may be defined in a eslint.config.js or in package.json depending on the setup.

Eslint is a tool that allows to find and fix problems in JavaScript code. The rules that are currently applied are the ones defined in the file `.eslintrc.js`.

## Exercise 2

Run `npm run serve` and open the app in your browser. Remember that npm looks at the `package.json` file (specially the `scripts` object) to find which command to execute.

```
npm run serve
```

## Exercise 3

The newly generated project contains a few placeholders. Cleanup your project so it does not contain neither useless assets, nor the hello world. In other words, delete `HelloWorld.vue`, its related assets and all its references. As at the end of each exercise, the vue cli should not report any error or warning.

to do that we have to delete the `HelloWorld.vue` file and the reference to it in the `App.vue` file.

## Exercise 4

Create the `HomePage` component inside the `pages` folder. Do not spend too much time on the template content, as it could be a simple sentence. Import it inside `App.vue`.

So you can find a file named `HomePage.vue` in the `pages` folder. I have also added a reference to it in the `App.vue` file. [HomePage.vue](#)

## Base layout

---

## Exercise 5

Let's begin with the root component, formally App (in src/App.vue). Replace its template with the following content and create the missing components. Add some content to the header (ex. fake home link, fake user name...) and legal credits to the footer. Eventually, polish the looks and feels with scoped CSS.

So we create 2 more file named [baseFooter.vue](#) and [baseHeader.vue](#) in the components folder. We also add a reference to them in the App.vue file.

## Questions 5

What is the difference between scoped and non-scoped CSS?

The difference between scoped and non-scoped CSS is that scoped CSS is only applied to the component where it is defined. Non-scoped CSS is applied to all the components.

## Exercise 6

In order to keep the root component App as simple as possible, extract everything related to the layout into a BaseLayout component. Using the slot API, allow BaseLayout to receive children (to be rendered between the header and the footer).

So we wrote a new file in [baseContent](#) to group the baseheader and the baseFooter.

## Question 6

How behaves non-prop attributes passed down to a component, when its template has a single root element? **Tips** : it is well documented by vue, but you can also try it yourself by passing the [style](#) attribute with a straight visual effect.

The non-prop attributes passed down to a component are added to the root element of the component. If the template has a single root element, the non-prop attributes are added to this element. If the template has multiple root elements, the non-prop attributes are added to the first root element.

# Exercice 7

---

Implement such a `BaseButton`, animated on hover and focus. Do not forget the disabled state. You may try these buttons on your `HomePage` for now.

We wrote a new file in `baseButton` and we added a reference to it in the `HomePage.vue` file.

# Exercice 8

---

Add the `color` prop to `BaseButton`. This prop accepts one of 'primary', 'warn' or 'danger' values. It defaults to primary and you should validate the given value matches the enum. Then, dynamically apply styles to the button based on that prop.

We modify the `baseButton` file to add the `color` prop and to apply the style based on the prop.

# Exercice 9

---

Add a button to the `HomePage` that is disabled for 2 seconds each time it is clicked. According to the above code, this just means the `@click` event listener attached to the instance of `AsyncComponent` instance returns a `Promise` that waits for 2 seconds before resolving. You can create such a `Promise` using its constructor and a `setTimeout`. Also, please write the event handler inside a dedicated method since it is a bit complex.

So we create a `AsyncButton` in the `HomePage` file. With this code :

```
handleClick() {
  setTimeout(() => {
    this.isPending = !this.isPending;
  }, 2000);
  this.isPending = !this.isPending;
},
```

Click me

Disabled Button

Warn Button

Danger Button

Async Button

Click me

Disabled Button

Warn Button

Danger Button

Loading ...Async Button

## Exercice 10

---

Change the behaviour of the previous button, so its waiting time increases by one second each it is clicked. Because AsyncButton waits for any promise, whatever how long it takes to resolve, you do not need and you should not change it. Instead, keep trace of the number of clicks in the internal state (data) of the HomePage component (see the counter app example) and use it while forging new promises.