Question 1
NPM has two install modes : local and global. The local mode is the default mode, and is used when we install a package in a project. The global mode is used when we install a package on the system, and is used to install tools like Vue CLI.

Because we want to use Vue CLI in any project, it is recommended to install it globally. Also, since Vue CLI may be heavy to install, it is recommended to install it globally to avoid re-downloading it multiple times and filling the disk. To install Vue CLI globally, we use the "npm install -g @vue/cli" command.

Exercice 1
To create a new Vue project called "vue-oauth-microsoft-graph", we first need to make sure that we are in an untracked folder. Then, we use the "vue create vue-oauth-microsoft-graph" command. An assistant will guide us through the creation of the project. We select Vue3 and its default configuration.

Vue CLI will automatically create a "vue-oauth-microsoft-graph" folder containing the project and setup all the dependencies. On top of that, Vue CLI will also create a git repository and commit the project. That is why it needs to be in an untracked folder (with no other git repository present).

Question 2
Webpack is required by Vue CLI because Vue uses multiple files to store its code, including proprietary files like ".vue" files. They need to be rendered into a JS file that the browser can understand. No browser can understand ".vue" files, because their conception is specific to Vue and not to JavaScript.

Question 3
Babel's role in a Vue project is to ensure that the code is compatible with all the wanted browsers. Babel will take each browser's characteristics into account, and will transpile the code to make it compatible with all of them.

You can however target a list of specific browsers, and Babel will only transpile the code to make it compatible with those browsers. The list of browsers can be seen in the "browserslist" section of the "package.json" file.

In our case, and by default, Vue CLI targets the following criteria in the package.json file:

```
"browserslist": [
    "> 1%",
    "last 2 versions",
    "not dead",
    "not ie 11"
]
```

As we can see, we can select browser based on a manual selection, but also with more relative criteria, like the market share. We can also exclude all browsers that are not supported anymore. In this case, both "not dead" and "not ie 11" criterias are used to exclude Internet Explorer 11.

Question 4
Eslint, like most of the tools named with "lint", is a tool that checks the code for errors and cleanliness. According to Wikipedia, lint is the name for the tiny bits of fiber and fluff shed by clothing, as the command should act like the lint trap in a clothes dryer, detecting small errors to great effect. Eslint will ensure our Vue code is clean, readable and consistent. It will also help us to avoid common mistakes.

Eslint is automatically installed by Vue CLI, and is configured in the "package.json" file (in our case). In the configuration, we inform eslint about our environment and the rules we want to apply. In our case, we tell eslint that we are using Vue3, and that we want to use the recommended rules, with no custom rules, and babel as the parser.

Exercice 2
To run the project, we use the "npm run serve" command. As we can see in the "package.json" file, this command is actually an alias for "vue-cli-service serve", "vue-cli-service build" and "vue-cli-service lint".

The project is built and served on the localhost:8080 address in development mode. We can see the new "Welcome to Your Vue.js App" project in the browser at this address.

Exercice 3
Now, we need to clean up the base project. We will delete the "HelloWorld.vue" in the components folder file as well as the logo.png file in the assets folder. Finally, we also need to remove all references from these files in the "App.vue" file.

We can replace the two lines in the "App.vue" file :

```
        <img alt="Vue logo" src="./assets/logo.png">
        <HelloWorld msg="Welcome to Your Vue.js App"/>
```

with a simple <h1> tag.

Next, we remove the "import HelloWorld from './components/HelloWorld.vue'" line, and we remove the HelloWorld component from the "components" section. Before committing our changes, we should re-run the "npm run serve" command to make sure that no errors were introduced. We can see our H1 tag in the browser, and no errors.

We can now commit our changes with the title "Ex 3: remove vue CLI placeholders».

Exercice 4
We create the src/pages folder and the HomePage.vue file in it. We make a simple <H1> tag in the HomePage.vue file. We import the HomePage.vue file in the App.vue file, declare the HomePage component in the "components" section, and replace the previous H1 tag with the HomePage component. The browser automatically refreshes and we can see our new H1 tag.

We can now commit our changes with the title "Ex 4: create HomePage component".

Exercice 5
This exercise has been committed in the "Ex 5: create BaseHeader and BaseFooter" commit.

Question 5
There are two ways to use CSS in Vue. The first one is scoped CSS, which is the default easiest way to use CSS in Vue. You just set your CSS code in the <style> tag of your component, and it will only apply to this component. The second way is non-scoped CSS, which is a global CSS file. You can create a CSS file in the src folder, and import it in the main.js file. This CSS file will then be applied to all the components.

Exercice 6
This exercise has been committed in the "Ex 6: create BaseLayout that uses slot API" commit.

Question 6
When a template has a single root element, non-prop attributes passed down to a component passes them to the root element. We can set a style "color: red;" on our HomePage component in the App.vue, and it will be applied to the root element <H1>.

Exercice 7
This exercise has been committed in the "Ex 7: create BaseButton with primary color" commit.

Exercice 8
This exercise has been committed in the "Ex 8: color palette and prop for BaseButton" commit.

Exercice 9
This exercise has been committed in the "Ex 9: add AsyncButton" commit.

Exercice 10
This exercise has been committed in the "Ex 10: slowing down the button on click" commit.

Question 7
The callback passed to .finally() is executed once the promise is completed, regardless of the result. We use it to reset the button state, even if the promise has encountered an error.
That is why we do not use the .then() callback, which only occurs in case of success, nor the .catch() callback which only occurs in case of an error.

Question 8
If we set inheritAttrs to true, our function is executed twice because the v-bind:onClick parameter is passed both to the AsyncButton component itself and to the handleClick() function in the AsyncButton component. We only want our function to be executed once, in the handleClick() function. We therefore need to prevent the passing of the v-bind:onClick parameter to the AsyncButton component itself, by setting inheritAttrs to false.