Compte Rendu

Nom : Hugo Wendjaneh - Fouad Id Gouahmane - Charles Raymondière **Date** : 07-03-2025 **Sujet** : Développement web

Question 1

> **Différence entre installation locale et globale avec npm ?**

- **Locale** (`npm install <package>` sans `-g`) : installe dans `node_modules/` du projet, et enregistre la dépendance dans `package.json`.
  - **Usage** : librairies et outils de build spécifiques au projet.
- **Globale** (`npm install -g <package>`) : installe dans un dossier partagé système, expose la commande dans le `PATH`.
  - **Usage** : CLI utilisables sur tous les projets.

Question 2

> **Pourquoi Webpack est-il nécessaire pour gérer à la fois plusieurs fichiers JS et les extensions `.vue` ?**

- **Modules** : regroupe plusieurs fichiers JavaScript modulaires (`import`/`export`) en un ou plusieurs fichiers de sortie.
- **Loaders / plugins** : transforme les Single File Components (`.vue`), les fichiers CSS/SCSS, les images...
- **Optimisation** : hot-reload, code splitting, tree-shaking, minification, injection de styles scoped...

Question 3

> **Rôle de Babel et influence de `browserslist` sur sa sortie**

- **Babel** : transpileur ESNext vers ES5+ pour assurer la compatibilité navigateurs.
- **browserslist** (dans `package.json`) : définit les cibles (ex. `"> 1%"`, `"last 2 versions"`, `"not dead"`).
  - Babel active/désactive des plugins selon les fonctionnalités supportées par ces cibles (ex. `async/await`, `modules`).

Question 4

> **Qu'est-ce qu'ESLint et quelles règles sont appliquées par défaut ?**

- **ESLint** : linter JavaScript permet de détecter les bugs, incohérences de style, mauvaises pratiques.
- **Dans Vue CLI** : via `@vue/cli-plugin-eslint` et le preset par défaut (règles Vue + standard JS).
  - Configurable dans `.eslintrc.js`, active des règles pour Vue (nomenclature, interpolation, directives) et JS (semi, quotes, indentation).

Question 5

> **Différence entre CSS scoped et non-scoped**

- `<style scoped>` : Vue génère un attribut unique sur chaque élément du composant, et scope ses règles à ce composant.
- **Sans** `scoped` : le style est global tout sélecteur s'applique à tous les éléments correspondants dans l'application. Cela peut aboutir à de potentiels conflits.

Question 6

> **Comment se comportent les attributs non-prop passés à un composant à un seul root ?**

- Par défaut, **tous** les attributs non déclarés en `props` sont appliqués **automatiquement** à l'élément racine du composant.

Question 7

> **Analyse du fonctionnement d'AsyncButton et choix de `.finally()`**

1. Le parent attache `@click="maMéthode"` sur `<AsyncButton>`.
2. Vue stocke ce listener dans `this.$attrs.onClick`.
3. `AsyncButton` :
    - `inheritAttrs: false`, empêche la propagation automatique.
    - Bind manuellement `:disabled="isPending"` et `@click.stop.prevent="handleClick"`.
    - Dans `handleClick()` :
        1. `this.isPending = true`
        2. Appel de la Promise d'origine → `const p = originalOnClick()`
        3. `p.finally(() => this.isPending = false)`
4. **Pourquoi `.finally()`** et non `.then()` :
    - `.finally()` s'exécute **toujours**, que la Promise soit résolue ou rejetée, garantissant le retour à l'état "non pending" même en cas d'erreur.

Question 8

> **Quel bug survient si `inheritAttrs: false` est absent ou vrai dans AsyncButton ?**

- **Sans `inheritAttrs: false`** :
    - Les attributs "non props" (`@click`, `disabled`, `class`, etc.) seraient automatiquement appliqués au `<base-button>` racine.
    - **Conséquences** :
        - L'écouteur original et le wrapper `handleClick` seraient tous deux attachés. Cela doublerais l'invocation ou l'ordre d'exécution non contrôlé.
        - Le `disabled` serait peut-être mal géré, brisant la logique de désactivation pendant la Promise.
- En le désactivant, on maîtrise exactement quels attributs et écouteurs sont relayés via `v-bind="$attrs"` ou manuellement.