T1d - Utilisation de dépendances

Motivation

Satisfait du résultat précédent, l'enseignant souhaite encore l'améliorer.
Il aimerait afficher le nom du groupe avec des grandes lettres, du type :

```
  ____                                  _
 / ___|_ __ ___  _   _ _ __   ___  / |
| |  _| '__/ _ \| | | | '_ \ / _ \ | |
| |_| | | | (_) | |_| | |_) |  __/ | |
 \____|_|  \___/ \__,_| .__/ \___| |_|
                      |_|
```

Un collègue lui a dit que pour faire cela, il pouvait utiliser le package **figlet** en l'ajoutant en tant que dépendance du projet.

Ajout de la dépendance

- Dans **VS Code**, ouvrez la vue **Terminal**
- Exécutez la commande :
  `pnpm add figlet`

Observation du résultat

Dossier node-modules

Dans la vue **Explorer** de **VS Code**, observez qu'un nouveau dossier est apparu :
`node_modules`

C'est dans ce dossier que sont installées les dépendances ajoutées au projet.
Dans un projet réel, il peut contenir de très nombreux fichiers et devenir très volumineux.

Fichier package.json

Ouvrez le fichier `package.json` pour observer son contenu
Une propriété supplémentaire lui a été ajoutée :

```
"dependencies": {
  "figlet": "^1.11.4"
}
```

Cette propriété donne la liste des dépendances du projet. Actuellement, il y en a une seule. C'est **figlet**.

La version indiquée pour **figlet**, n'est pas forcément la version installée, mais la version minimale que le projet attend. **figlet** doit donc être en version 1.11.4 ou supérieure

Licence Creative Commons BY-SA 4.0 - AMBLARD Emmanuel 2026

Fichier pnpm-lock.yaml

Ouvrez le fichier `pnpm-lock.yaml`

Il s'agit d'un nouveau fichier qui vient d'être créé. Il indique ce qui a été réellement installé.

Dans la section **packages** (à la fin), on voit qu'il y a en fait deux dépendances qui ont été installées :

commander@14.0.3
figlet@1.11.4

figlet ne fonctionne pas tout seul. Il a besoin du package **commander**

Les numéros de versions sont ceux qui sont réellement installés. figlet est donc en version 1.11.4

Utilisation de figlet

- Ouvrez le fichier **groupe1.js**
- À la fin du code, insérez une ligne vide et tapez :
  `const titre = figlet`
  Puis utilisez l'autocomplétion pour terminer la saisie.
- Observez l'instruction qui a été ajoutée au début du fichier :
  `import figlet from 'figlet'`
  Cette instruction importe un objet **figlet** qui donne accès aux fonctions fournies par le package **figlet**
- Finalisez d'écrire la ligne que vous avez commencé à taper pour qu'elle devienne :
  `const titre = figlet.textSync('Groupe 1')`
- Ajoutez l'instruction qui affiche le contenu de la variable `titre`
- Exécutez le script pour vérifier son bon fonctionnement.

Affichage en couleur

Pour mettre une dernière touche à son script, l'enseignant souhaite que le nom du groupe soit affiché en couleur.

Il a vu qu'il existait un package nommé **chalk** qui permet d'obtenir ce résultat.

- Ajoutez `chalk` en tant que dépendance du projet.
  Utilisez bien **pnpm** et non **npm**, car **pnpm** est plus optimisé.
- Modifiez l'instruction d'affichage pour que `titre` soit affiché en **rouge**.
  `console.log(chalk.red(titre))`
  Pensez à faire l'importation nécessaire.
- Pour voir la couleur, il faut exécuter le script dans la vue **Terminal**.
  Si vous utilisez **Run Code**, le résultat s'affiche dans la vue **Output** qui ne gère pas les couleurs.

Licence Creative Commons BY-SA 4.0 - AMBLARD Emmanuel 2026

Utilisation d'un script

- Ouvrez le fichier `package.json`
- Observez que `chalk` a été ajoutée en tant que dépendance du projet.
- Observez également la présence d'une section `scripts.`
  Elle est là depuis le début.

La section `scripts` permet d'enregistrer des lignes de commande un peu longues qu'on utilise fréquemment.

Actuellement, elle contient un exemple nommé `test` qui affiche un message, puis simule une erreur en exécutant `exit 1`.

- Exécutez le script `test` :
  ```
  pnpm run test
  ```
  Deux lignes sont affichées correspondant aux 2 actions du script `test`, dont la ligne en rouge qui est produite par `exit 1`
- Quand il n'y a pas de confusion possible entre le nom du script et une commande de **PNPM**, on peut omettre le mot `run`
  ```
  pnpm test
  ```

Création d'un script

- Dans la section `scripts`, ajoutez un item nommé `g1` qui exécute `groupe1.js`
  ```
  "g1": "node groupe1.js"
  ```
  Pensez à mettre une virgule à la fin de l'item précédent, afin de respecter la syntaxe **JSON**.
- Exécutez votre script :
  `pnpm run g1` ou `pnpm g1`

Remarque En pratique, la création d'un script est surtout utile lorsqu'elle évite de taper une longue ligne de commande ou d'avoir à mémoriser un code compliqué.

Licence Creative Commons BY-SA 4.0 - AMBLARD Emmanuel 2026