T1c - Modules JavaScript

Création d'un module

Diagnostic

Un sujet inquiète l'enseignant. Il intervient dans plusieurs classes et doit donc gérer plusieurs groupes d'élèves.

Il envisage de créer autant de fichiers que nécessaire : **groupe1.js**, **groupe2.js, groupe3.js**, …

Chaque fichier contiendra un tableau `eleves`. Mais il aimerait **éviter** que le code qui calcule et affiche les moyennes soit **dupliqué** dans chacun des fichiers.

La solution consiste à placer ce code dans un **module** JavaScript.

CommonJS et ES Modules

vue d'ensemble

La modularité nécessite d'adopter un **standard** qui définit les règles que doit suivre un paquet logiciel pour pouvoir être considéré comme un **module**. Ainsi, il est réutilisable dans diverses situations.

CommonJS est le standard le plus ancien (2009). C'est celui qui a été adopté à l'origine par **Node.js**.

Plus récemment (2015) est apparu le standard **ESM** ou **ES Modules** (**ECMAScript Modules**). C'est la solution préconisée, car **ESM** est plus flexible et plus optimisé.

Comment les reconnaître ?

CommonJS

Sur Internet, on trouve beaucoup d'exemples de code qui sont conformes à **CommonJS**.

On les reconnaît au fait qu'ils utilisent la fonction **require()** :

```
const http = require("http");
const fs = require('fs');
```

Cette technique est donc à **éviter**.

ES Modules

On reconnaît un code qui applique le standard **ESM** au fait qu'il utilise le mot-clé **import**.

```
import { createApp } from 'vue'
import App from './App.vue'
```

C'est la technique **à utiliser**. On peut l'employer aussi bien dans un navigateur web qu'avec **Node.js**

Licence Creative Commons BY-SA 4.0 - AMBLARD Emmanuel 2026

Configuration de Node.js pour utiliser ES Modules

Vue d'ensemble

La règle appliquée par **Node.js** a changé au cours du temps :

- Avant sa version **22**, **Node.js** utilisait **CommonJS** par défaut et il fallait le configurer pour qu'il utilise **ESM**.
- Depuis sa version **22** (2024), **Node.js** est capable de détecter automatiquement s'il doit utiliser **CommonJS** ou **ESM**. La configuration pour **ESM** est donc facultative, mais si on la fait, cela améliore les performances, car cela évite à **Node** d'avoir à faire la détection automatique.

En conclusion, lorsqu'on veut utiliser le standard **ESM**, il vaut mieux effectuer la configuration même avec une version récente de **Node.js**

Création du fichier package.json

- Dans **VS Code**, faites afficher la vue **Terminal** : Ctrl+ù
- Exécutez la commande :

  ```
  pnpm init
  ```

- Dans la vue **Explorer** (à gauche), observez qu'il y a un nouveau fichier nommé `package.json`. Il s'agit du fichier de configuration de votre projet pour **Node.js**

  Ouvrez ce fichier pour voir son contenu.

Comme vous le voyez, il s'agit d'un fichier au format **JSON**.

Notez que les noms des propriétés sont entourés par des **doubles quotes**. C'est obligatoire dans un fichier **JSON**.

Pour le moment, la majorité de son contenu n'a pas une grande utilité. La plupart des informations contenues ont seulement un rôle documentaire. En effet, ce fichier permet de :

- Donner un nom au projet ainsi qu'un n° de version
- Écrire une description
- Indiquer son auteur (vous pouvez mettre votre nom dans la propriété **author**)
- etc.

Observez la dernière propriété :

```
"type": "module"
```

C'est elle qui indique à **Node.js** qu'il doit utiliser le standard **ESM** dans ce projet.

Module pour le calcul des moyennes

Fichier moyenne.js

- Par copier-coller, dupliquez le fichier `groupe1.js` de façon à créer un nouveau fichier nommé `moyenne.js`

Licence Creative Commons BY-SA 4.0 - AMBLARD Emmanuel 2026

- Ouvrez le fichier **moyenne.js**

  L'idée est de n'y conserver que le code qui sera commun à tous les groupes.
  Pour cela, suivez les consignes ci-dessous.

- À la fin du fichier, ajoutez la déclaration d'une fonction nommée **afficher()** :

```
function afficher( eleves ) {
}
```

- Dans le fichier, repérez les blocs d'instructions suivants et déplacez-les dans la fonction **afficher()** :
  - Le bloc qui ajoute une propriété **moyenne** à chacun des éléments du tableau **eleves**
  - Le bloc qui fait afficher le prénom et la moyenne de chaque élève.

À présent, dans le fichier, il y a :
- 3 fonctions : **Eleve()**, **getMoyenne()** et **afficher()**
- du code qui se trouve en dehors de ces fonctions

- Supprimez tout le code qui se trouve en dehors des fonctions.

  Vous pouvez conserver la directive **'use strict"**

Ainsi, le fichier **moyenne.js** ne contient plus que les 3 fonctions. Il est prêt à être transformé en module **ESM**.

Transformation en module ESM

- Ajoutez le mot-clé **export** devant chacune des 3 fonctions

  Exemple :

```
export function Eleve(prenom, notes) {
```

À présent, vous avez un module conforme au standard **ESM**.

Le mot-clé **export** permet d'indiquer les fonctions qui pourront être appelées depuis l'extérieur du module.

Utilisation du module

Configuration de l'autocomplétion de code

Pour que **VS Code** génère les instructions **import** lorsque vous utiliserez l'autocomplétion, il a besoin de la présence d'un fichier **jsconfig.json**.

Ce fichier indique que son dossier correspond à la racine d'un projet **JavaScript**.

- Créez un nouveau fichier nommé **jsconfig.json**
- Mettez-y le contenu suivant :

Licence Creative Commons BY-SA 4.0 - AMBLARD Emmanuel 2026

```
{
  "compilerOptions": {
    "module": "nodenext"
  }
}
```

Cela indique à **VS Code** que vous utilisez **Node.js** dans une version récente.

C'est important pour qu'il formate correctement les instructions `import`.
C'est-à-dire en y incluant les extensions des fichiers **JavaScript**.

Suppression du code inutile dans le fichier groupe1.js

- Dans le fichier **groupe1.js**, conservez uniquement :
  - La directive `'use strict'`
  - La déclaration du tableau `eleves`

  Supprimez tout le reste du code, y compris la fonction **Eleve()**.
  Il doit donc rester moins de 10 lignes.

Erreur d'exécution

À ce stade, si on cherche à exécuter le script **groupe1.js** on obtient une erreur, car la fonction **Eleve()** n'est plus définie dans ce fichier.

Pour qu'il n'y ait plus d'erreur, il suffit d'importer la fonction **Eleve()**.

Import du module

- Dans le fichier **groupe1.js**, placez le curseur juste à la fin du mot **Eleve**
- Appuyez sur les touches **Ctrl**+**Espace** comme pour faire l'autocomplétion
  Sélectionnez **Eleve**
- Observez qu'une instruction `import` a été générée.
  `import { Eleve } from "./moyenne.js"`
- Puis, sur une ligne vide à la fin, tapez le début du mot : **afficher**
  Utilisez l'autocomplétion pour terminer la saisie.
- Observez ce qui s'est passé.
  VS Code n'a pas créé une nouvelle instruction **import**.
  Il a modifié l'instruction existante qui est devenue :
  `import { afficher, Eleve } from './moyenne.js'`
- À la fin du fichier, complétez l'instruction afin que la fonction **afficher()** reçoive en paramètre le tableau `eleves`.
- Exécutez le script pour vérifier qu'il affiche bien les moyennes des élèves.

Licence Creative Commons BY-SA 4.0 - AMBLARD Emmanuel 2026

```
Aline 13
Brice 15
Cathy 9
Diego 11
```

Licence Creative Commons BY-SA 4.0 - AMBLARD Emmanuel 2026