*Note Js*

**N02**

**Activities**

Sommaire

**[Activité 1 — “Diagnostic Machine” 2](#_Toc20823)**

[Discussion 2](#_Toc14921)

**[Activité 2 — “Explorateur de fichiers”: 3](#_Toc11396)**

**[Activité 3 —“Moniteur d’événements” 3](#_Toc7118)**

[Discussion : 3](#_Toc21614)

**[Activité 4 — “Classe de Journalisation (Logger)” : 4](#_Toc19907)**

[Discussion : 4](#_Toc5138)

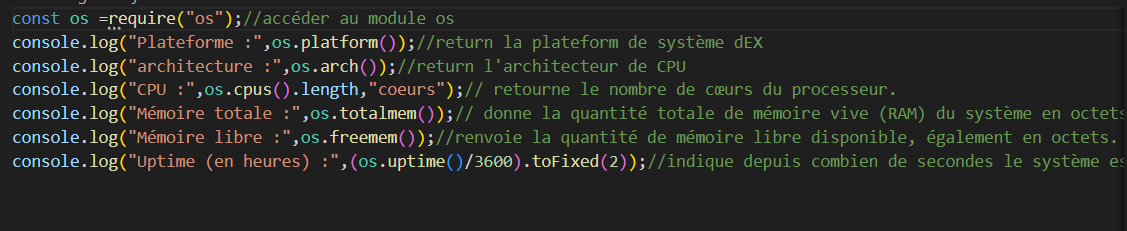
**[Activité 5 — “Serveur Node minimaliste” : 5](#_Toc5741)**

**[Activité 6 — “Combinaison finale” : 6](#_Toc2804)**

[Discussion : 6](#_Toc2119)

# Activité 1 — “Diagnostic Machine”

*Code:*



Discussion ***:***

1. *Quelle est la différence entre os.platform() et os.arch() ?*

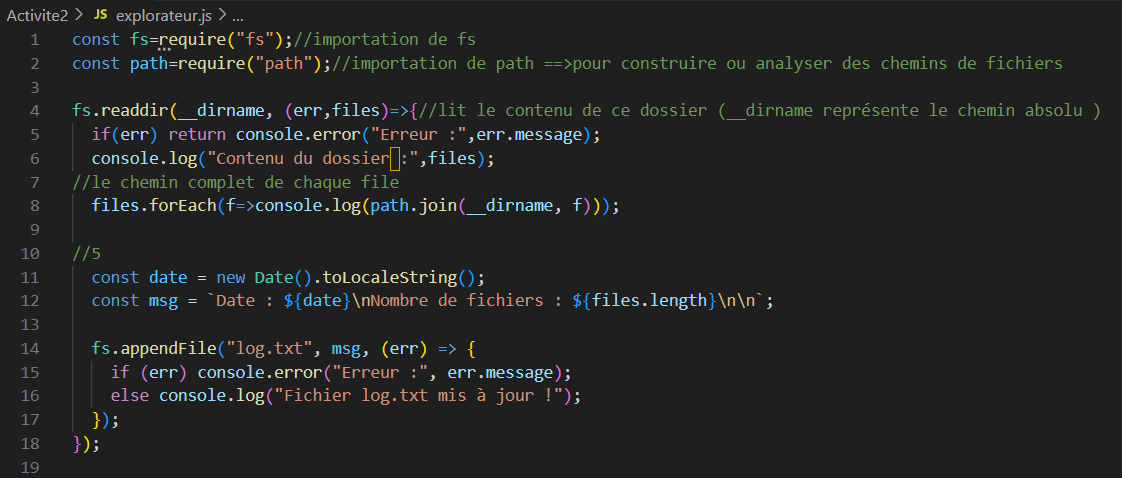
* ***os.platform():*** *Décrit le type de système d’exploitation*
* ***os.arch() :*** *Décrit le type de processeur*

1. *À quoi pourrait servir cette information dans une application réelle (ex : tableau de bord*

*système) ?*

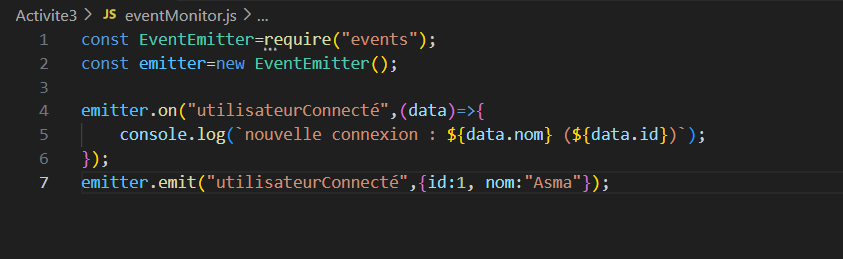
*Ces informations peuvent être utiles dans une application réelle, par exemple dans un tableau de bord système pour afficher les caractéristiques du serveur. Elles permettent aussi d’adapter le comportement d’un programme selon la plate-forme, de surveiller les performances, ou de vérifier la compatibilité avant une installation.*

# Activité 2 — “Explorateur de fichiers”:



# Activité 3 —“Moniteur d’événements”

*Code:*



## Discussion :

1. *Que se passe-t-il si l’écouteur est enregistré après l’émission de l’événement ?*

* *il ne sera pas déclenché*

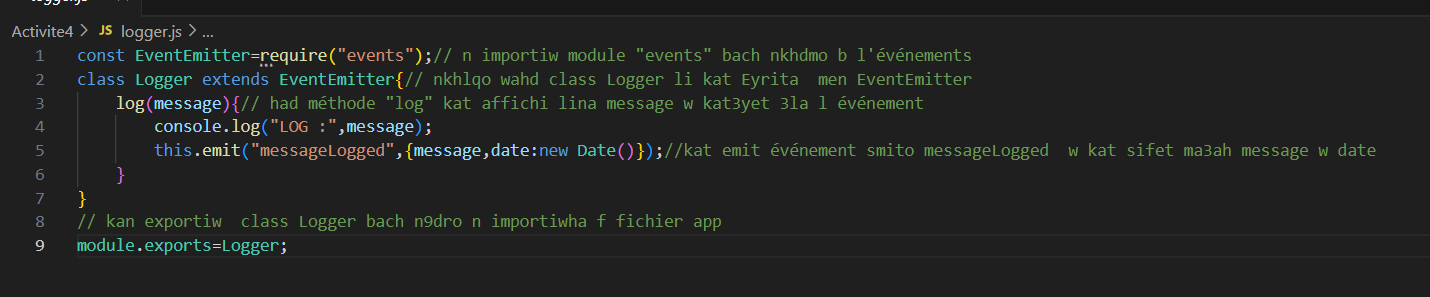
1. *Peut-on avoir plusieurs écouteurs pour un même événement ?*

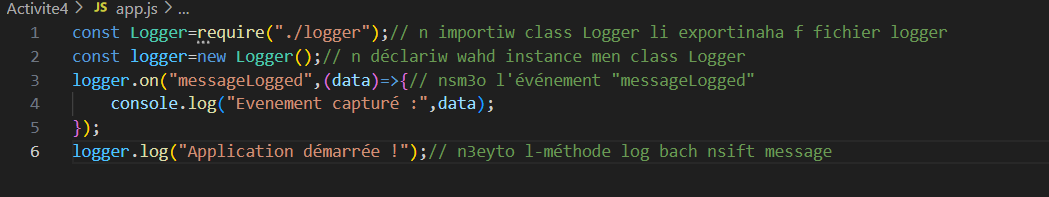
* *Oui, on peut avoir plusieurs écouteurs pour un même événement.*

# 

# Activité 4 — “Classe de Journalisation (Logger)” :

*Code*





## Discussion :

1. *Quelle est la différence entre une instance directe d’EventEmitter et une classe qui l’étend ?*

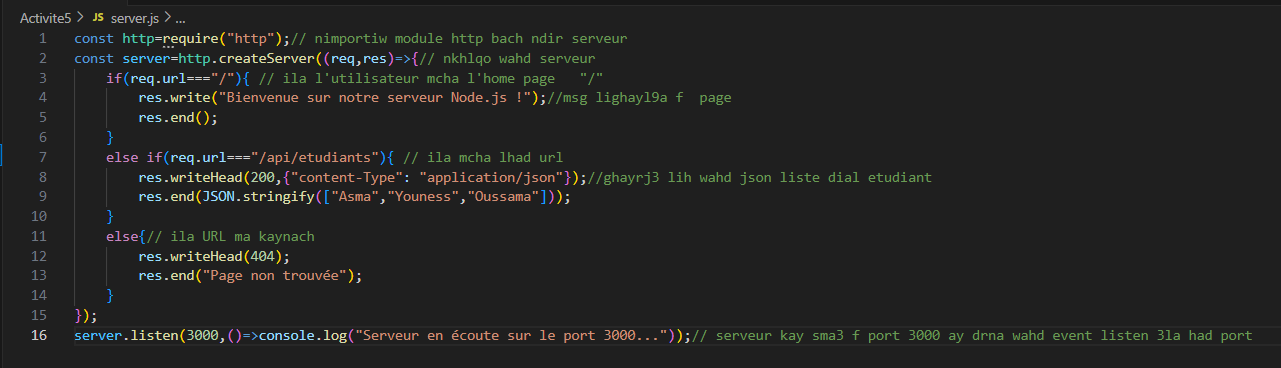
* *****Instance directe:*** *simple est rapide, juste pour émettre/écouter.***
* *****Classe qui étend EventEmitter:*** *on peut créer un objet avec ses propres méthodes qui émet des événements***

1. *Pourquoi encapsuler la logique dans une classe ?*

* la classe permet de **structurer le code et de le rendre réutilisable**.

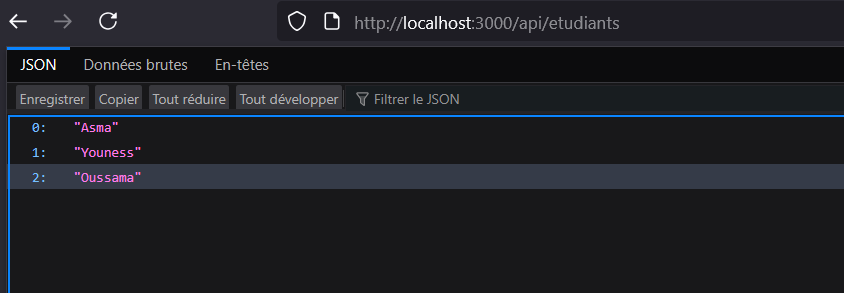
# Activité 5 — “Serveur Node minimaliste” :

Code:



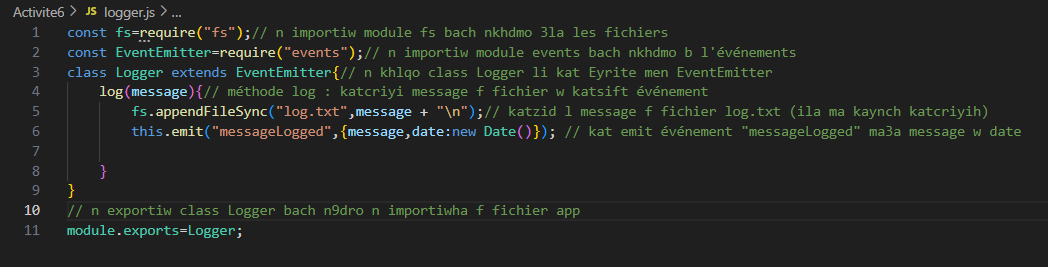
Affichage:

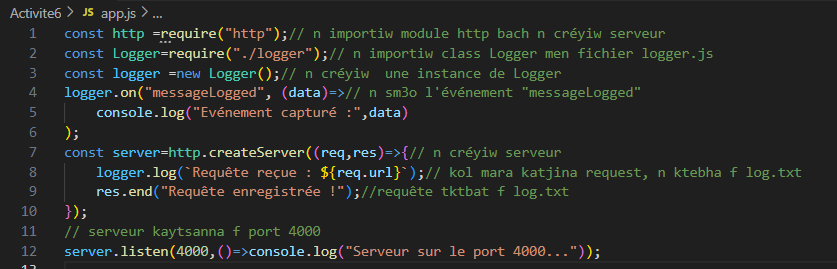
## 



# Activité 6 — “Combinaison finale” :

Code:





## Discussion :

1. *Que signifie “non-bloquant” dans le contexte du module fs ?*

* *“****Non bloquant****” dans le module* ***fs*** *signifie que l’opération ne bloque pas l’exécution du programme. Node.js continue d’exécuter le reste du code pendant que la lecture ou l’écriture du fichier s’effectue en arrière-plan.*

1. *Comment les événements permettent-ils de découpler les modules?*

* *Un module peut simplement émettre un événement, et un autre module peut l’écouter sans que le premier ait besoin de le connaître. Ça rend le code plus facile à maintenir.*

1. *Pourquoi un serveur HTTP Node peut-il gérer des milliers de connexions avec un seul thread ?*

* *Un serveur HTTP Node peut gérer des milliers de connexions avec un seul thread grâce à son fonctionnement asynchrone.*

1. *Quelle serait la prochaine étape (ex: Express, middlewares, JSON parsing) ?*