

TP n°2 Javascript avancé

Avant de commencer, n'oubliez pas que la documentation de Javascript est disponible [ici](#) ou [là](#). Nous utiliserons [CodeSandbox](#), un environnement en ligne pour coder et exécuter du Javascript.

Exercice 1 :

Ecrire un script utilisant la bibliothèque [Chalk](#) qui écrit chaque élément du tableau suivant dans la couleur de l'élément.

```
const colors = ["blue", "red", "green", "yellow", "cyan"]
```

⚠ `const chalk = require('chalk');` est l'ancienne syntaxe, il faut utiliser `import chalk from "chalk"`.

⚠ N'oubliez pas d'ajouter la bibliothèque `Chalk` avec le bouton `Add dependency` de CodeSandbox.

Exercice 2 :

Ecrire un script qui récupère 100 users via [randomuser](#) avec [axios](#) et qui les affichent.

- En utilisant les [promesses](#)
- En utilisant [async/await](#)

⚠ Attention à l'aspect asynchrone du code ! Mettez des logs un peu partout et regarder l'ordre d'affichage.

⚠ N'oubliez pas d'ajouter la bibliothèque `Axios` avec le bouton `Add dependency` de CodeSandbox.

Exercice 3 :

Ecrire un script qui récupère 1000 users et qui affiche les users dont la timezone est Paris. (Utiliser [filter\(\)](#))

- En utilisant les [promesses](#)
- En utilisant [async/await](#)

Exercice 4 :

Ecrire un script qui récupère 1000 users et qui stocke dans un second tableau uniquement le prénom et le nom de tous les users dont le timezone est Paris. (Utiliser [filter\(\)](#), [map\(\)](#) et [forEach\(\)](#))

- En utilisant les [promesses](#)
- En utilisant [async/await](#)

Exercice 5

La fonction [setTimeout\(\)](#) permet d'exécuter du code après x ms de façon asynchrone. Malheureusement, à l'époque de la création de cette fonction, les promesses n'existaient pas !

Créer une fonction wrapper `setTimeout()` dans une promesse puis écrire un programme qui affiche le dialogue suivant :

```
- toc toc
**attendre 500 millisecondes**
- qui est là?
**attendre 10 secondes**
- C'est Internet Explorer
```

- En utilisant [setTimeout\(\)](#)

- En utilisant les [promesses](#) et votre nouvelle fonction
- En utilisant [async/await](#) et votre nouvelle fonction

Autres ressources

- [Comprendre les promesses en js](#) [FR]
- [Javascript event loop explained](#) [EN]
- [Concurrence et boucle d'événements](#) [FR]
- [Async/await](#) [FR]