Asynchrone, setTimeout & setInterval

Touchet Laurent / Afec 2025

Sommaire

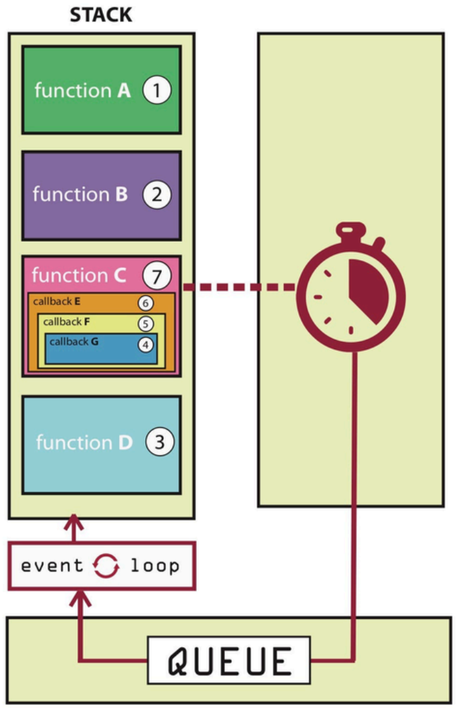
1. **Comment fonctionne JavaScript normalement ?**
2. **C’est quoi une action asynchrone ?**
3. **Exemples d’actions asynchrones**
4. **Introduction à setTimeout**
5. **Introduction à setInterval**
6. **Ce qu’il faut retenir**

Comment fonctionne JavaScript normalement ?

JavaScript exécute le code ligne par ligne, dans l'ordre.

Chaque instruction doit être terminée avant de passer à la suivante.

Exemple : Si une tâche est lente, tout le reste est bloqué !



Pourquoi l'asynchrone est nécessaire ?

Sur un site web, certaines opérations prennent plusieurs secondes :

- Télécharger une image
- Charger des données d'un serveur
- Lire un fichier local

Si JavaScript attendait à chaque fois, le site deviendrait inutilisable pour l'utilisateur !

C’est quoi une action asynchrone ?

- Une action asynchrone est une tâche qui se lance mais n’empêche pas le reste du code de s’exécuter.
- JavaScript continue son travail et revient plus tard quand la tâche est finie.
- Résultat : le site reste fluide même si certaines tâches prennent du temps.

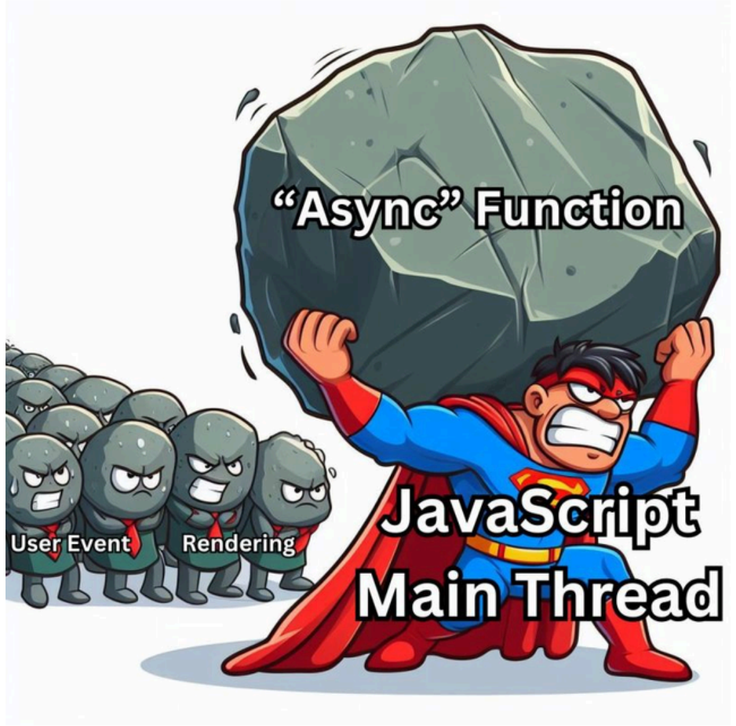
La métaphore du gâteau au four

- On prépare un gâteau et on le met au four (30 minutes).
- Pendant ce temps, on ne peut pas rester pas devant le four, on fait autre chose.
- En JS, c’est pareil : on continue le programme sans attendre.

Exemples d’actions asynchrones

Toutes ces actions doivent être gérées sans bloquer la page :

- Appeler une API (ex: récupérer la météo)
- Charger une image / vidéo sur un site
- Lire un fichier stocké sur un serveur
- Attendre la réponse d’un formulaire



Comment JavaScript gère ça ?

- **Callbacks** : fonction appelée plus tard.
- **Promises** : promesse d’avoir une réponse.
- **Async/Await** : écrire l’asynchrone comme si c’était du code normal.

Introduction à setTimeout

setTimeout permet de programmer une action pour qu'elle se fasse plus tard, après un certain temps (en millisecondes).

```
setTimeout(function() {
 console.log("Hello après 2 secondes !");
}, 2000)
```

Pourquoi c'est utile ?

- Quand on veut attendre avant de faire quelque chose.
- Exemple concret : afficher un message après un chargement, lancer une animation après 2 secondes, afficher une pop-up après un délai, etc...

Introduction à setInterval

setInterval() permet de répéter une fonction toutes les X millisecondes.

```
setInterval(function() {
 console.log("Hello toutes les 3 secondes !");
}, 3000);
```

Pourquoi c’est utile ?

- Quand on veut faire quelque chose plusieurs fois automatiquement, sans recliquer.
- Exemple concret : créer une horloge en temps réel, faire bouger une animation, vérifier régulièrement un serveur, faire un chronomètre…

Ce qu’il faut retenir

- JavaScript est **synchrone** de base.
- **L’asynchrone** permet de continuer l’exécution du site sans être bloqué.
- **L’asynchrone** est utile pour travailler avec les serveurs, les images ou les données “lentes”.