🔍 Qu'est-ce que `useEffect` en React ?

`useEffect` est un **Hook** qui permet d'exécuter du code après le rendu d'un composant. Il est souvent utilisé pour **gérer les effets de bord** comme :
✔️ **Les appels API** (fetch des données).
✔️ **L'écoute d'événements** (ex: ajouter un `eventListener`).
✔️ **L'interaction avec le DOM** (ex: modifier le `title` de la page).
✔️ **Le nettoyage des ressources** (ex: fermer une connexion WebSocket).

📌 Syntaxe de base

```
import { useEffect } from "react";

useEffect(() => {
    // Code exécuté après le rendu du composant
});
```

💡 Ce code est exécuté **après que React ait affiché le composant dans le navigateur**.

📌 Exemple 1 : Modifier le titre de la page après le rendu

```
import { useEffect } from "react";

function TitrePage() {
    useEffect(() => {
        document.title = "Bienvenue sur mon site !";
    });

    return <h1>Regarde l'onglet du navigateur 👀</h1>;
}
```

➡️ Ici, `document.title` est mis à jour chaque fois que le composant est affiché.

📌 useEffect et le tableau de dépendances

Le **deuxième argument** de `useEffect` est un **tableau de dépendances** qui contrôle **quand** l'effet doit être exécuté.

1️⃣ Sans tableau de dépendances : s'exécute après chaque rendu

```
useEffect(() => {
    console.log("Ce code s'exécute après CHAQUE rendu !");
});
```

💡 À **chaque fois** que le composant est affiché ou mis à jour, le code est exécuté.

2️⃣ Avec `[]` : s'exécute UNE seule fois (au premier rendu seulement)

```
useEffect(() => {
    console.log("Ce code s'exécute UNE SEULE FOIS !");
}, []);
```

💡 L'effet ne s'exécute **qu'une seule fois**, après le premier rendu du composant.

3️⃣ Avec des dépendances `[state, props]` : s'exécute UNIQUEMENT quand elles changent

```
const [count, setCount] = useState(0);

useEffect(() => {
    console.log(`Le compteur est maintenant à ${count}`);
}, [count]);
```

💡 L'effet **ne s'exécute que lorsque `count` change**.

📌 Exemple 2 : Récupérer des données depuis une API

```
import { useEffect, useState } from "react";

function ListeUtilisateurs() {
    const [users, setUsers] = useState([]);

    useEffect(() => {
        fetch("https://jsonplaceholder.typicode.com/users")
            .then(response => response.json())
            .then(data => setUsers(data));
    }, []);

    return (
        <ul>
            {users.map(user => (
                <li key={user.id}>{user.name}</li>
            ))}
        </ul>
    );
}
```

➡️ L'effet (`useEffect`) est exécuté **une seule fois** (`[]` en dépendance).
➡️ `fetch` récupère la liste des utilisateurs et met à jour `users` avec `setUsers(data)`.

📌 Nettoyer un effet avec useEffect

Quand un composant est supprimé (unmounted), `useEffect` peut **exécuter une fonction de nettoyage**.

📌 Exemple : Ajouter et retirer un eventListener

```
import { useEffect } from "react";

function EcouteurClavier() {
    useEffect(() => {
        const handleKeyPress = (event) => {
            console.log(`Touche pressée : ${event.key}`);
        };

        window.addEventListener("keydown", handleKeyPress);

        // 🔥 Nettoyage : supprimer l'eventListener quand le composant
disparaît
        return () => {
            window.removeEventListener("keydown", handleKeyPress);
        };
    }, []);

    return <h1>Appuie sur une touche et regarde la console !</h1>;
}
```

➡️ Au **montage**, `useEffect` ajoute `handleKeyPress`.
➡️ Au **démontage**, la fonction `return` enlève l'`eventListener` pour éviter les bugs.

📌 Résumé

Variante	Exécution
`useEffect(() => { ... });`	Après chaque rendu (chaque mise à jour)
`useEffect(() => { ... }, []);`	**Une seule fois**, après le premier rendu
`useEffect(() => { ... }, [count]);`	**À chaque changement de `count`**
`useEffect(() => { return () => { ... } }, []);`	Nettoyage au démontage du composant

🚀 **Conclusion** ✅ `useEffect` est un outil puissant pour exécuter du code après un rendu.
✅ Il est essentiel pour **les appels API, la manipulation du DOM et les abonnements**.

✅ Le **tableau de dépendances** permet de **contrôler quand l'effet doit être déclenché**.
✅ Il faut **nettoyer les effets** si nécessaire pour éviter les fuites mémoire.
