useEffect

useEffect

Permet d'exécuter des effets de bord (appels API, timers, listeners...) après le rendu.

Syntaxe

```
useEffect(() => {
  // effet
  return () => { /* cleanup */ }; // optionnel
}, [/* dépendances */]);
```

Les 3 modes selon le tableau de dépendances

```
// 1. Sans tableau → s'exécute après CHAQUE render
useEffect(() => {
  console.log("render");
});

// 2. Tableau vide → s'exécute UNE FOIS au montage
useEffect(() => {
  console.log("monté");
}, []);

// 3. Avec dépendances → s'exécute quand une dépendance change
useEffect(() => {
  console.log("userId a changé :", userId);
}, [userId]);
```

Exemple complet : fetch de données

```
function UserProfile({ userId }) {
  const [user, setUser] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    setLoading(true);
    const controller = new AbortController();

    fetch(`/api/users/${userId}`, { signal: controller.signal })
      .then(res => res.json())
      .then(data => { setUser(data); setLoading(false); })
      .catch(err => { if (err.name !== 'AbortError') console.error(err); });

    return () => controller.abort();
  }, [userId]);

  if (loading) return <p>Chargement...</p>;
  return <p>{user?.nom}</p>;
}
```

Pièges courants

- **Dépendances manquantes** : toujours inclure toutes les variables utilisées dans l'effet
- **Boucle infinie** : si une dépendance est un objet/tableau créé dans le render, il change à chaque fois → utiliser `useMemo` ou le déplacer hors du composant
- **State stale** : préférer la forme fonctionnelle de setState dans un useEffect

---

Revision #1
Created 29 March 2026 12:57:34 by Hugo
Updated 29 March 2026 12:57:34 by Hugo