TP React 6
`useEffect` et cycle de vie

Objectifs pédagogiques

À la fin de ce TP, vous serez capable de :
— comprendre le rôle du hook `useEffect`
— distinguer les différents moments d'exécution d'un effet
— exécuter du code au chargement d'un composant
— réagir à un changement de state

Introduction

Dans une application React, certaines actions ne doivent pas être exécutées à chaque affichage de l'interface :
— chargement de données
— affichage de messages
— communication avec une API
Le hook `useEffect` permet de gérer ces **effets secondaires**.

> `useEffect` permet d'exécuter du code à des moments précis du cycle de vie d'un composant.

1 Importer le hook `useEffect`

Dans `src/App.jsx`, importez les hooks nécessaires :

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

2 Effet exécuté au chargement

Exemple : afficher un message au chargement du composant.

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

function App() {
  useEffect(() => {
    console.log("Composant chargé");
  }, []);

  return (
    <div>
      <h1>Test useEffect</h1>
    </div>
  );
}

export default App;
```

Le tableau vide `[]` indique que l'effet est exécuté une seule fois, au chargement du composant.

3 Effet dépendant d'un state

Ajoutons un compteur et observons son évolution.

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

function App() {
  const [compteur, setCompteur] = useState(0);

  useEffect(() => {
    console.log("Le compteur a changé :", compteur);
  }, [compteur]);

  return (
    <div>
      <p>Compteur : {compteur}</p>
      <button onClick={() => setCompteur(compteur + 1)}>
        Incrémenter
      </button>
    </div>
  );
}

export default App;
```

L'effet est exécuté uniquement lorsque la valeur `compteur` change.

4 Effet exécuté à chaque rendu

Il est possible d'exécuter un effet à chaque rendu du composant.

```
useEffect(() => {
  console.log("Rendu du composant");
});
```

Un effet sans tableau de dépendances s'exécute à chaque rendu. À utiliser avec précaution.

5 Résumé des comportements

— `useEffect(..., [])` : au chargement
— `useEffect(..., [state])` : au changement du state
— `useEffect(...)` : à chaque rendu

Conclusion

Dans ce TP, vous avez appris à :
— utiliser le hook `useEffect`
— comprendre le cycle de vie d'un composant
— déclencher des actions au bon moment

`useEffect` permet à React d'interagir avec le monde extérieur.