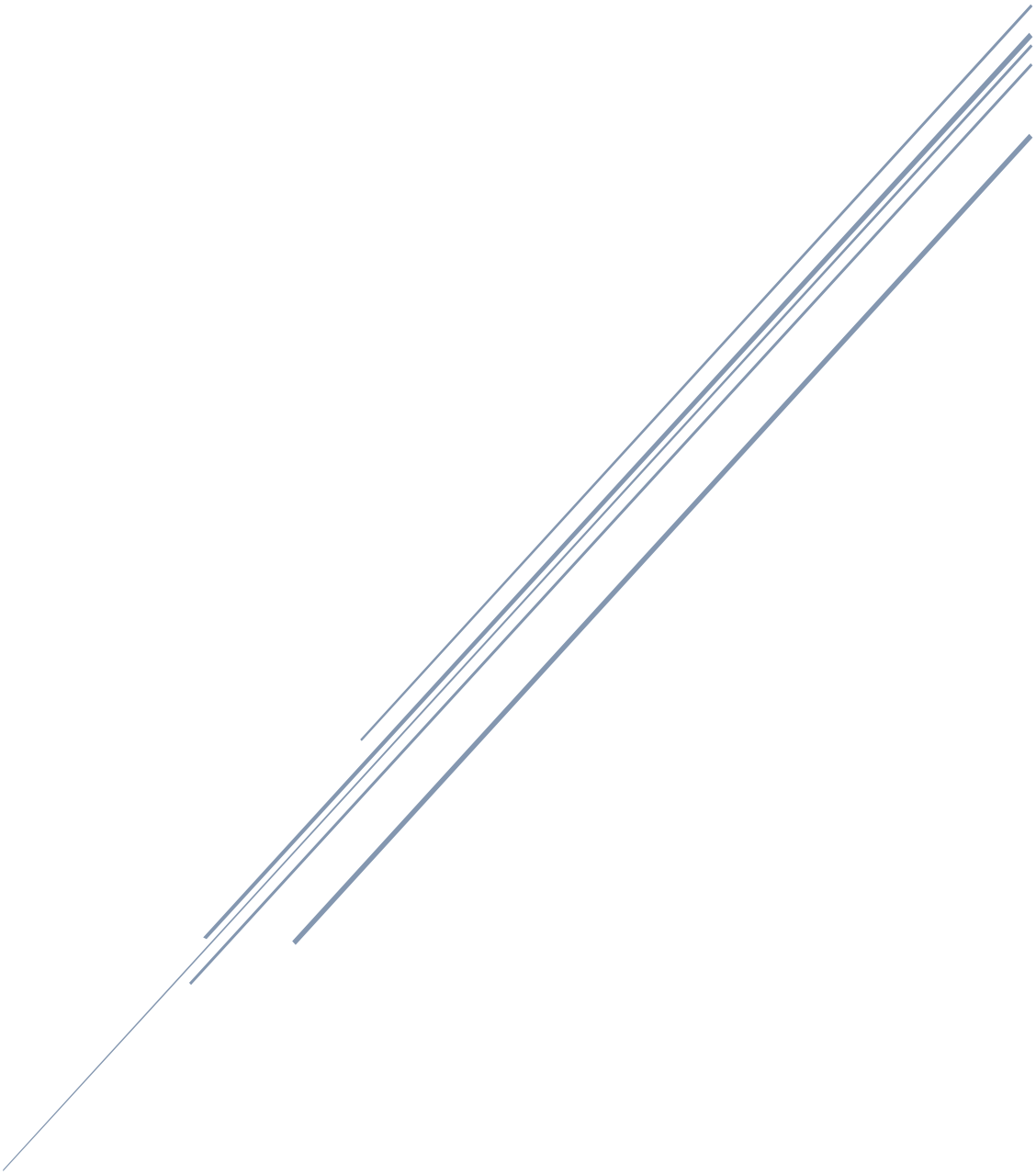
MAITRISE DES HOOKS REACT : USESTATE ET USEEFFECT

Gestion des état par les composants fonctionnels.



ISMO

Chapitre 09 – Hooks useSate & useEffect de React

MAHDI KELLOUCH

Table des matières

Introduction aux Hooks : Transformez vos composants fonctionnels

Les Hooks, introduits avec React 16.8, permettent d'utiliser l'état local et d'autres fonctionnalités de React sans écrire de classes. Les deux Hooks les plus utilisés sont :

- **useState** : pour gérer l'état local dans un composant fonctionnel.
- **useEffect** : pour gérer les effets de bord (ex. : appels API, timers).

I. useState : Gérer l'état dans un composant fonctionnel

1. Déclaration et fonctionnement

useState est utilisé pour créer une variable d'état dans un composant fonctionnel.

```
import React, { useState } from 'react';
function Counter() {
  const [count, setCount] = useState(0); // Initialisation de l'état à 0

  return (
    <div>
      <p>Le compteur est à : {count}</p>
      <button onClick={() => setCount(count + 1)}>Incrémenter</button>
    </div>
  );
}
export default Counter;
```

2. Exemple : Application de liste de tâches

Créez une application où l'utilisateur peut ajouter et supprimer des tâches.

Code :

```
import React, { useState } from 'react';

function TodoApp() {
  const [tasks, setTasks] = useState([]);
  const [task, setTask] = useState('');

  const addTask = () => {
    if (task.trim()) {
      setTasks([...tasks, task]);
      setTask('');
    }
  };
  const removeTask = (index) => {
    const updatedTasks = tasks.filter((_, i) => i !== index);
    setTasks(updatedTasks);
  };
```

```
  return (
    <div>
      <h1>Liste de Tâches</h1>
      <input
        type="text"
        value={task}
        onChange={(e) => setTask(e.target.value)}
        placeholder="Nouvelle tâche"
      />
      <button onClick={addTask}>Ajouter</button>
      <ul>
        {tasks.map((t, i) => (
          <li key={i}>
            {t} <button onClick={() => removeTask(i)}>Supprimer</button>
          </li>
        ))}
      </ul>
    </div>
  );
}
export default TodoApp;
```

Exercice d'application

1. Créez une application **React** où un utilisateur peut ajouter des notes (comme un journal).
2. Ajoutez un bouton pour réinitialiser toutes les notes.

II. useEffect : Gestion des effets de bord

1. Déclaration et fonctionnement

useEffect permet d'effectuer des actions secondaires comme :

- Appels à une API,
- Mise à jour du DOM,
- Gestion de timers.

```
import React, { useState, useEffect } from 'react';

function Timer() {
  const [seconds, setSeconds] = useState(0);

  useEffect(() => {
    const interval = setInterval(() => {
      setSeconds((prev) => prev + 1);
    }, 1000);

    return () => clearInterval(interval); // Nettoyage
  }, []);
```

```
  return <p>Le temps écoulé : {seconds} secondes</p>;
}
export default Timer;
```

2. Exemple : Application météo (API Fetch)

Une application qui affiche la météo d'une ville en utilisant une API (OpenWeatherMap par exemple).

Code :

```
import React, { useState, useEffect } from 'react';

function WeatherApp() {
  const [city, setCity] = useState('Paris');
  const [weather, setWeather] = useState(null);

  useEffect(() => {
    fetch(`https://api.openweathermap.org/data/2.5/weather?q=${city}&ap-
pid=YOUR_API_KEY&units=metric`)
      .then((response) => response.json())
      .then((data) => setWeather(data));
  }, [city]);

  return (
    <div>
      <h1>Application Météo</h1>
      <input type="text" value={city}onChange={(e) => setCity(e.target.value)}
        placeholder="Entrer une ville"
      />
      {weather && (
        <div>
          <h2>{weather.name}</h2>
          <p>Température : {weather.main.temp} °C</p>
        </div>
      )}
    </div>
  );
}
export default WeatherApp;
```

Exercice d'application

1. Modifiez l'application météo pour afficher une liste des villes récemment consultées.
2. Ajoutez une fonctionnalité pour effacer l'historique des recherches.

III. Exercices pratiques

Exercice 1 : Compteur avec historique

Créez un compteur qui affiche les valeurs précédentes dans une liste.

Exercice 2 : Chronomètre

Créez un chronomètre avec démarrage, arrêt et réinitialisation. Gérez le timer avec useEffect.

Exercice 3 : Liste des produits

Créez une application de gestion de produits avec les fonctionnalités suivantes :

- Ajout de produits,
- Filtrage par catégorie.

IV. Questions à choix multiples (QCM)

1. Quelle est la valeur initiale renvoyée par useState ?

a) Un tableau contenant la valeur initiale et une fonction.

b) Une fonction seule.

c) Une valeur vide.

d) Une valeur null.

2. Quand useEffect est-il exécuté ?

a) Après chaque rendu.

b) Avant le premier rendu.

c) Seulement quand le composant est démonté.

d) À la demande de l'utilisateur.

3. Quelle méthode nettoie les effets dans useEffect ?

a) clearEffect.

b) Une fonction de retour.

c) componentWillUnmount.

d) resetEffect.

4. Que signifie l'absence d'un tableau de dépendances dans useEffect ?

- a) Il ne s'exécutera jamais.
- b) Il s'exécutera à chaque mise à jour du composant.
- c) Il s'exécutera une seule fois.
- d) Cela génère une erreur.

Conclusion

- **useState** est idéal pour gérer des états locaux simples.
- **useEffect** permet de gérer les effets secondaires et les actions asynchrones.
- Leur combinaison est puissante pour créer des applications réactives et dynamiques.