TP React 4
State et interactions (`useState`)

Objectifs pédagogiques

À la fin de ce TP, vous serez capable de :
— comprendre la notion de state en React
— utiliser le hook `useState`
— gérer des interactions utilisateur
— mettre à jour dynamiquement l'interface

Introduction

Jusqu'à présent, l'interface affichait des données fixes. Le **state** permet de stocker des données qui peuvent évoluer.

Quand le state change, React met automatiquement à jour l'interface.

1 Importer le hook `useState`

Dans `src/App.jsx`, importez le hook :

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

2 Créer un state simple : compteur

Exemple : un compteur qui augmente au clic.

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

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

  return (
    <div>
      <p>Compteur : {compteur}</p>

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

export default App;
```

Explication :

- `compteur` : valeur actuelle
- `setCompteur` : fonction de mise à jour
- `useState(0)` : valeur initiale

3 Gestion des événements

React propose des événements similaires à JavaScript :

- `onClick`
- `onChange`
- `onSubmit`

En React, on passe une fonction à l'événement : `onClick={...}`.

4 Afficher / masquer un texte

Ajoutez un second exemple qui modifie l'affichage selon un état.

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

function App() {
  const [visible, setVisible] = useState(true);

  return (
    <div>
      <button onClick={() => setVisible(!visible)}>
        Afficher / Masquer
      </button>

      {visible && <p>Ce texte peut être masqué.</p>}
    </div>
  );
}

export default App;
```

Le rendu conditionnel permet d'afficher un élément seulement si une condition est vraie.

5 Règles importantes sur le state

— Ne jamais modifier directement le state
— Toujours utiliser la fonction de mise à jour
— Un changement de state déclenche un nouveau rendu (re-render)

Un state doit rester simple et avoir un rôle clair.

Conclusion

Dans ce TP, vous avez appris à :
— utiliser `useState`
— gérer des interactions utilisateur
— rendre l'interface dynamique

Le state est le cœur des applications React interactives.