TP : Le Compteur Interactif

Objectif : Transformer une page statique en application dynamique.

Concepts clés : Le Hook useState (l'état) et les événements (onClick).

Fichier de travail : src/App.tsx

Contexte

Vous venez de créer votre "Hello World". Maintenant, nous allons rendre cette page interactive. React ne met pas à jour l'écran automatiquement si vous changez une variable classique (let ou const). Il faut utiliser une variable spéciale appelée **State**.

Instructions pas à pas

Étape 1 : Importer le Hook

Tout en haut de votre fichier src/App.tsx, nous devons importer l'outil qui permet de créer un état.

Action : Ajoutez l'import nécessaire en haut du fichier.

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

Étape 2 : Initialiser le State

Dans votre fonction App, vous devez déclarer votre compteur. Rappelez-vous que useState renvoie un tableau avec deux éléments : la valeur actuelle et la fonction pour la modifier.

Action : Complétez la ligne suivante au début de la fonction App :

```
function App() {
  // TODO : Déclarez le state 'count' et la fonction 'setCount'
  // Initialisez le compteur à 0
  const [___, ___] = useState(0);

  // ...
```

Étape 3 : Créer la logique

Juste sous la déclaration du state (et avant le return), créez les deux fonctions qui seront

appelées lors du clic.

Action : Complétez le corps des fonctions :

```
const increment = () => {
  // TODO : Utilisez setCount pour ajouter 1 à la valeur actuelle
  ___
};

const decrement = () => {
  // TODO : Utilisez setCount pour retirer 1 à la valeur actuelle
  ___
};
```

Étape 4 : Mettre à jour l'interface (JSX)

Nous allons afficher la valeur du compteur et lier nos fonctions aux boutons.

Action : Remplacez le return et complétez les trous (___) :

```
return (
  <div style={{ textAlign: 'center', marginTop: '50px', fontFamily: 'Arial' }}>
    <h1>Compteur Interactif</h1>

    {/* TODO : Affichez la variable du state ici */}
    <p style={{ fontSize: '40px', fontWeight: 'bold' }}>
      {___}
    </p>

    <div style={{ display: 'flex', gap: '10px', justifyContent: 'center' }}>
      {/* TODO : Liez la fonction decrement au clic */}
      <button onClick={___} style={{ padding: '10px 20px' }}>
        - Diminuer
      </button>

      {/* TODO : Liez la fonction increment au clic */}
      <button onClick={___} style={{ padding: '10px 20px' }}>
        + Augmenter
      </button>
    </div>
  </div>
);
```

Étape 5 (Bonus) : Le Style Conditionnel

Ajoutons de la logique visuelle. Si le compteur passe en dessous de zéro, le texte doit devenir rouge.

Action : Utilisez une condition ternaire pour changer la couleur.

```
<p style={{
  fontSize: '40px',
  fontWeight: 'bold',
  // TODO : Si count est inférieur à 0, la couleur doit être 'red', sinon 'black'
  color: ___ ? 'red' : 'black'
}}>
  {count}
</p>
```

Structure Globale à compléter

Voici le squelette de votre fichier pour vérifier que vous avez tout placé au bon endroit.

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

function App() {
  // 1. Déclaration du State
  const [___, ___] = useState(0);

  // 2. Fonctions de gestion (Handlers)
  const increment = () => {
    ___
  };

  const decrement = () => {
    ___
  };

  // 3. Rendu (JSX)
  return (
    <div>
      {/* ... Votre JSX avec les événements onClick ... */}
    </div>
  );
```

```
}

export default App;
```