Exercice 4 - Compteur interactif

Objectif

Créer un composant interactif qui réagit aux clics de l'utilisateur.

Concepts abordés

- Hook `useState`
- Gestionnaires d'événements (`onClick`)
- Mise à jour du state
- Re-render automatique

Contexte

Vous créez un compteur pour une application. Ce compteur doit pouvoir être incrémenté, décrémenté et remis à zéro.

Énoncé

Étape 1 : Compteur simple

Créer un composant `Compteur` avec :

- Un affichage de la valeur actuelle
- Un bouton "+1" qui incrémente
- Un bouton "-1" qui décrémente

Étape 2 : Ajouter un bouton Reset

Ajouter un bouton "Reset" qui remet le compteur à 0.

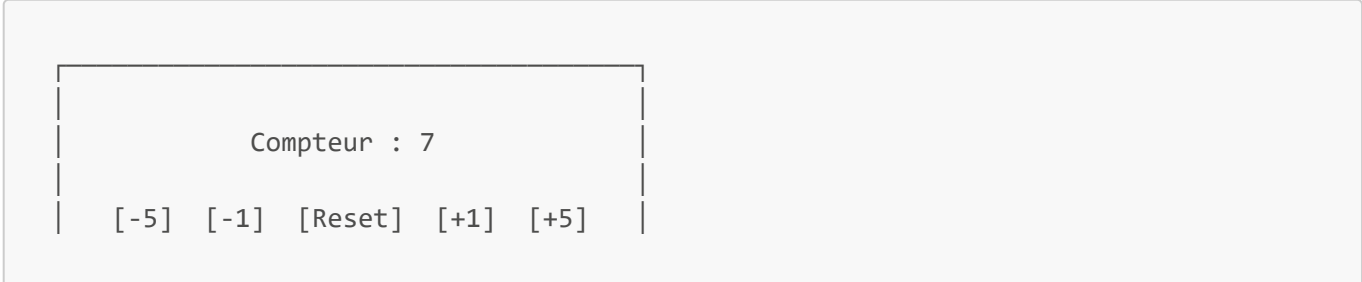
Étape 3 : Empêcher les valeurs négatives

Le compteur ne doit pas descendre en dessous de 0.

Étape 4 : Ajouter un pas configurable

Ajouter des boutons +5 / -5 en plus de +1 / -1.

Résultat attendu



```
│                                          │
└──────────────────────────────────────────┘
```

Indices

▶ Indice 1 : Syntaxe de useState

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

function Compteur() {
  const [count, setCount] = useState(0);
  //      │        │              │
  //      │        │              └── Valeur initiale
  //      │        └── Fonction pour modifier
  //      └── Valeur actuelle
}
```

▶ Indice 2 : Gérer un clic

```jsx
function handleClick() {
  setCount(count + 1);
}

<button onClick={handleClick}>+1</button>

// Ou directement inline
<button onClick={() => setCount(count + 1)}>+1</button>
```

▶ Indice 3 : Empêcher les valeurs négatives

```jsx
function decrement() {
  if (count > 0) {
    setCount(count - 1);
  }
}

// Ou avec Math.max
setCount(Math.max(0, count - 1));

// Ou désactiver le bouton
<button disabled={count === 0}>-1</button>
```

▶ Indice 4 : Forme fonctionnelle de setState

```
// Recommandé quand la nouvelle valeur dépend de l'ancienne
setCount(prevCount => prevCount + 1);
```

Points d'attention

Importer useState

```
// ERREUR : useState n'est pas défini
function Compteur() {
  const [count, setCount] = useState(0);  // ReferenceError!
}

// CORRECT : import en haut du fichier
import { useState } from 'react';

function Compteur() {
  const [count, setCount] = useState(0);  // OK
}
```

Passer une fonction, pas l'appeler

```
// ERREUR : la fonction est exécutée immédiatement au rendu !
<button onClick={handleClick()}>  // NON : handleClick()

// CORRECT : on passe la référence de la fonction
<button onClick={handleClick}>    // OUI : handleClick (sans parenthèses)

// CORRECT : fonction fléchée qui appelle la fonction
<button onClick={() => handleClick()}>  // OUI avec ()
<button onClick={() => increment(5)}>   // Pour passer des arguments
```

Le state est asynchrone

```
function handleClick() {
  setCount(count + 1);
  console.log(count);  // Affiche l'ANCIENNE valeur !
}

// Solution : utiliser la forme fonctionnelle
function handleClick() {
  setCount(prev => {
    console.log(prev + 1);  // Nouvelle valeur
    return prev + 1;
  });
}
```

Forme fonctionnelle : quand l'utiliser ?

```
// Simple : quand la nouvelle valeur ne dépend pas de l'ancienne
setCount(0);
setCount(42);

// Fonctionnelle : quand elle dépend de l'ancienne (RECOMMANDÉ)
setCount(prev => prev + 1);
setCount(prev => prev - 1);
setCount(prev => prev * 2);
```

Correction

```
// src/exercices/ex04-compteur/Compteur.jsx

import { useState } from 'react';

function Compteur() {
  // ═══════════════════════════════════════════════════════════
  // DÉCLARATION DU STATE
  //
  // useState retourne un tableau avec 2 éléments :
  // - count : la valeur actuelle (commence à 0)
  // - setCount : la fonction pour modifier cette valeur
  //
  // Quand setCount est appelé, React re-rend le composant
  // avec la nouvelle valeur de count.
  // ═══════════════════════════════════════════════════════════
  const [count, setCount] = useState(0);

  // ═══════════════════════════════════════════════════════════
  // HANDLERS (fonctions de gestion des événements)
  //
  // On utilise la forme fonctionnelle (prev => ...) pour être sûr
  // d'avoir la dernière valeur, même en cas de clics rapides.
  // ═══════════════════════════════════════════════════════════

  function increment(pas = 1) {
    setCount(prev => prev + pas);
  }

  function decrement(pas = 1) {
    // Math.max empêche de descendre sous 0
    setCount(prev => Math.max(0, prev - pas));
  }

  function reset() {
```

```
    setCount(0);
  }

  // ═══════════════════════════════════════════════════════════════
  // RENDU
  //
  // Chaque bouton a un onClick qui appelle la fonction appropriée.
  // Pour passer un argument (le pas), on utilise une fonction fléchée.
  // ═══════════════════════════════════════════════════════════════
  return (
    <div className="compteur">
      {/* Affichage de la valeur */}
      <h2>Compteur : {count}</h2>

      {/* Contrôles */}
      <div className="compteur-controls">
        {/*
          Pour -5 et -1, on désactive si on ne peut pas décrémenter
          autant sans passer sous 0
        */}
        <button
          onClick={() => decrement(5)}
          disabled={count < 5}
        >
          -5
        </button>

        <button
          onClick={() => decrement(1)}
          disabled={count === 0}
        >
          -1
        </button>

        <button onClick={reset}>
          Reset
        </button>

        {/*
          Pour +1 et +5, pas de fonction fléchée nécessaire
          si on utilise une fonction avec paramètre par défaut
        */}
        <button onClick={() => increment(1)}>
          +1
        </button>

        <button onClick={() => increment(5)}>
          +5
        </button>
      </div>

      {/* Message conditionnel */}
      {count === 0 && (
        <p className="message">Le compteur est à zéro</p>
```

```
      )}

      {count >= 10 && (
        <p className="message">Bravo ! Vous avez atteint {count} !</p>
      )}
    </div>
  );
}

export default Compteur;
```

Version simplifiée (sans pas configurable)

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

function CompteurSimple() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <h2>Compteur : {count}</h2>

      <button onClick={() => setCount(prev => Math.max(0, prev - 1))}>
        -1
      </button>

      <button onClick={() => setCount(0)}>
        Reset
      </button>

      <button onClick={() => setCount(prev => prev + 1)}>
        +1
      </button>
    </div>
  );
}
```

CSS associé (optionnel)

```
/* src/exercices/ex04-compteur/Compteur.css */

.compteur {
  text-align: center;
  padding: 2rem;
}

.compteur h2 {
  font-size: 2rem;
  margin-bottom: 1rem;
```

```
}

.compteur-controls {
  display: flex;
  gap: 0.5rem;
  justify-content: center;
}

.compteur-controls button {
  padding: 0.5rem 1rem;
  font-size: 1rem;
  border: 1px solid #ddd;
  border-radius: 4px;
  cursor: pointer;
  background: white;
}

.compteur-controls button:hover:not(:disabled) {
  background: #f0f0f0;
}

.compteur-controls button:disabled {
  opacity: 0.5;
  cursor: not-allowed;
}

.message {
  margin-top: 1rem;
  color: #666;
  font-style: italic;
}
```

Schéma mental : Cycle du State

```
┌─────────────────────────────────────────────────────────────┐
│                                                             │
│  1. INITIALISATION                                          │
│     useState(0)  →  count = 0                               │
│                                                             │
│  2. AFFICHAGE                                               │
│     <h2>Compteur : {count}</h2>  →  "Compteur : 0"          │
│                                                             │
│  3. INTERACTION                                             │
│     Clic sur "+1"  →  onClick déclenché                     │
│                                                             │
│  4. MISE À JOUR                                             │
│     setCount(prev => prev + 1)  →  count = 1                │
│                                                             │
│  5. RE-RENDER                                               │
│     React re-exécute le composant avec count = 1            │
```

```
│     <h2>Compteur : {count}</h2>  →  "Compteur : 1"                  │
│                                                                     │
│  6. RETOUR À L'ÉTAPE 3 (attente interaction)                        │
│                                                                     │
└─────────────────────────────────────────────────────────────────────┘
```

Pour aller plus loin

1. Ajouter un input pour choisir le pas d'incrémentation
2. Sauvegarder la valeur dans le localStorage
3. Créer plusieurs compteurs indépendants

```jsx
// Input pour le pas
const [pas, setPas] = useState(1);

<input
  type="number"
  value={pas}
  onChange={(e) => setPas(Number(e.target.value))}
  min="1"
/>
<button onClick={() => increment(pas)}>+{pas}</button>
```