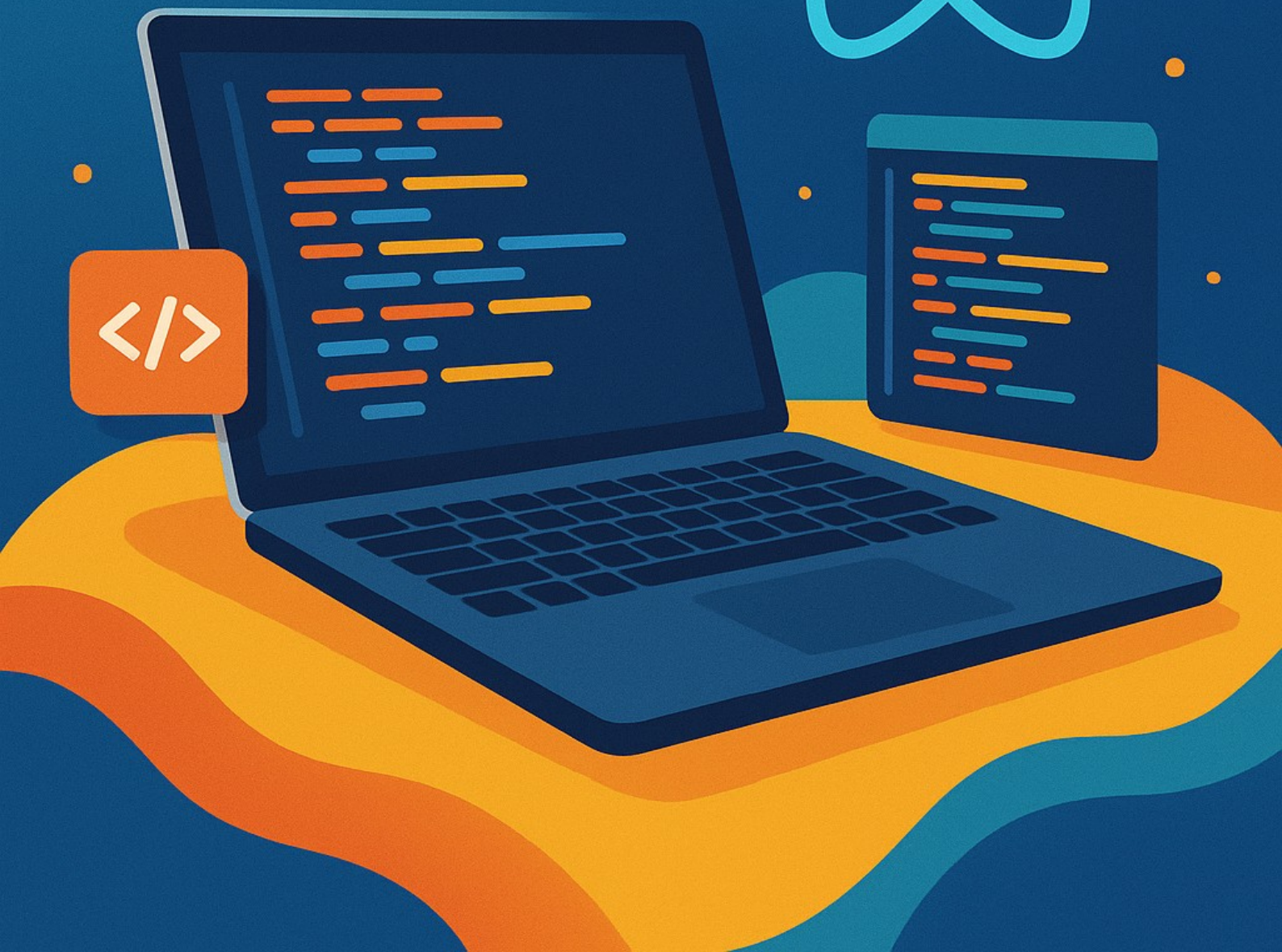
# FORMATION REACT

## NIVEAU INTERMÉDIAIRE



# React JS Niveau Intermédiaire

*Projets Concrets et Applications Réelles*

**⚡ Hooks et Interactivité ⚡**

**Partie 1 : Application de Compteur (Pages 1-15)**

# 📚 Ce que nous allons créer

Dans cette première partie, nous allons créer une **Application de Compteur Interactive** avec plusieurs fonctionnalités :

- Compteur simple avec boutons +/-
- Compteur avec pas personnalisable
- Historique des modifications
- Styles dynamiques selon la valeur

## Table des Matières

# 1. Créer le Projet Compteur

Commençons par créer notre projet. Ouvrez votre terminal et tapez :

```
npx create-react-app compteur-app
cd compteur-app
npm start
```

Une fois le projet créé et démarré, vous verrez la page React par défaut. Modifions-la !

## Nettoyer le Projet

Ouvrez **src/App.js** et remplacez tout par ce code propre :

```
import './App.css';

function App() {
  return (
    <div className="App">
      <h1>Application de Compteur</h1>
    </div>
  );
}

export default App;
```

Maintenant, nettoyons les styles. Ouvrez **src/App.css** et remplacez par :

```
.App {
  text-align: center;
  min-height: 100vh;
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  color: white;
  font-family: 'Arial', sans-serif;
}

h1 {
```

```
  font-size: 3rem;
  margin-bottom: 2rem;
  text-shadow: 2px 2px 4px rgba(0,0,0,0.3);
}
```

> 💡 *Un bon dégradé, c'est comme une bonne pizza : ça met tout le monde d'accord !*

## 2. useState : Notre Premier Hook

useState est le hook le plus important ! Il permet à votre composant de se 'souvenir' de choses. Créons notre compteur !

### Compteur Basique

Modifiez **src/App.js** :

```
import { useState } from 'react';
import './App.css';

function App() {
  // Déclarer une variable d'état 'count' avec valeur initiale 0
  const [count, setCount] = useState(0);

  return (
    <div className="App">
      <h1>Application de Compteur</h1>

      <div className="compteur-display">
        <h2>{count}</h2>
      </div>

      <div className="boutons">
        <button onClick={() => setCount(count - 1)}>
          - Diminuer
        </button>

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

        <button onClick={() => setCount(count + 1)}>
          + Augmenter
        </button>
      </div>
    </div>
  );
}

export default App;
```

### Explications ligne par ligne :

- **useState(0)** : Crée une variable d'état avec 0 comme valeur initiale
- **count** : La valeur actuelle du compteur
- **setCount** : La fonction pour modifier la valeur
- **onClick** : Déclenche une fonction quand on clique

## Styliser les Boutons

Ajoutez ces styles dans **src/App.css** :

```
.compteur-display {
  background: rgba(255, 255, 255, 0.2);
  border-radius: 20px;
  padding: 40px 80px;
  margin: 30px 0;
  backdrop-filter: blur(10px);
  box-shadow: 0 8px 32px rgba(0, 0, 0, 0.3);
}

.compteur-display h2 {
  font-size: 6rem;
  margin: 0;
  font-weight: bold;
}

.boutons {
  display: flex;
  gap: 20px;
  margin-top: 30px;
}

button {
  background: white;
  color: #667eea;
  border: none;
  padding: 15px 30px;
  font-size: 1.2rem;
  font-weight: bold;
  border-radius: 10px;
  cursor: pointer;
  transition: all 0.3s ease;
  box-shadow: 0 4px 15px rgba(0, 0, 0, 0.2);
}

button:hover {
  transform: translateY(-3px);
  box-shadow: 0 6px 20px rgba(0, 0, 0, 0.3);
}

button:active {
```

```
  transform: translateY(0);
}
```

✨ **Sauvegardez et admirez votre compteur ! Cliquez sur les boutons et voyez la magie opérer.**

## 3. Fonctions et Événements

Améliorons notre code en créant des fonctions dédiées. C'est plus propre et professionnel !

### Refactoriser avec des Fonctions

```
import { useState } from 'react';
import './App.css';

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

  // Fonction pour augmenter
  const augmenter = () => {
    setCount(count + 1);
  };

  // Fonction pour diminuer
  const diminuer = () => {
    setCount(count - 1);
  };

  // Fonction pour réinitialiser
  const reinitialiser = () => {
    setCount(0);
  };

  return (
    <div className="App">
      <h1>Application de Compteur</h1>

      <div className="compteur-display">
        <h2>{count}</h2>
      </div>

      <div className="boutons">
        <button onClick={diminuer}>- Diminuer</button>
        <button onClick={reinitialiser}>🔄 Réinitialiser</button>
        <button onClick={augmenter}>+ Augmenter</button>
      </div>
    </div>
```

```
  );
}

export default App;
```

*💡 Des fonctions bien nommées, c'est comme des panneaux de signalisation : tout le monde sait où aller !*

## Ajouter un Pas Personnalisable

Ajoutons la possibilité d'augmenter/diminuer par 5 ou 10 :

```
import { useState } from 'react';
import './App.css';

function App() {
  const [count, setCount] = useState(0);
  const [pas, setPas] = useState(1); // Nouveau state pour le pas

  const augmenter = () => {
    setCount(count + pas);
  };

  const diminuer = () => {
    setCount(count - pas);
  };

  const reinitialiser = () => {
    setCount(0);
  };

  return (
    <div className="App">
      <h1>Application de Compteur</h1>

      <div className="compteur-display">
        <h2>{count}</h2>
      </div>

      {/* Sélecteur de pas */}
      <div className="pas-selector">
        <label>Pas d'incrémentation : </label>
        <select value={pas} onChange={(e) => setPas(Number(e.target.value))}>
          <option value={1}>1</option>
          <option value={5}>5</option>
          <option value={10}>10</option>
        </select>
      </div>

      <div className="boutons">
        <button onClick={diminuer}>- {pas}</button>
        <button onClick={reinitialiser}>🔄 Réinitialiser</button>
```

```
        <button onClick={augmenter}>+ {pas}</button>
      </div>
    </div>
  );
}

export default App;
```

Ajoutez ce style dans App.css :

```
.pas-selector {
  margin: 20px 0;
  font-size: 1.2rem;
}

.pas-selector select {
  margin-left: 10px;
  padding: 8px 15px;
  font-size: 1rem;
  border-radius: 5px;
  border: 2px solid white;
  background: rgba(255, 255, 255, 0.9);
  cursor: pointer;
}
```

# 4. Composants et Props

Rendons notre code plus modulaire en créant des composants réutilisables avec des props !

## Créer un Composant Bouton

Créez un nouveau fichier **src/BoutonCompteur.js** :

```
function BoutonCompteur({ onClick, children, couleur }) {
  return (
    <button
      onClick={onClick}
      style={{
        background: couleur || 'white',
        color: couleur ? 'white' : '#667eea'
      }}
    >
      {children}
    </button>
  );
}

export default BoutonCompteur;
```

Maintenant, utilisez-le dans App.js :

```
import { useState } from 'react';
import './App.css';
import BoutonCompteur from './BoutonCompteur';

function App() {
  const [count, setCount] = useState(0);
  const [pas, setPas] = useState(1);

  const augmenter = () => setCount(count + pas);
  const diminuer = () => setCount(count - pas);
  const reinitialiser = () => setCount(0);

  return (
    <div className="App">
      <h1>Application de Compteur</h1>
```

```
      <div className="compteur-display">
        <h2>{count}</h2>
      </div>

      <div className="pas-selector">
        <label>Pas : </label>
        <select value={pas} onChange={(e) => setPas(Number(e.target.value))}>
          <option value={1}>1</option>
          <option value={5}>5</option>
          <option value={10}>10</option>
        </select>
      </div>

      <div className="boutons">
        <BoutonCompteur onClick={diminuer} couleur="#e74c3c">
          - {pas}
        </BoutonCompteur>

        <BoutonCompteur onClick={reinitialiser} couleur="#95a5a6">
          🔄 Reset
        </BoutonCompteur>

        <BoutonCompteur onClick={augmenter} couleur="#27ae60">
          + {pas}
        </BoutonCompteur>
      </div>
    </div>
  );
}

export default App;
```

*💡 Les props, c'est comme passer des ingrédients à un chef : vous donnez ce qu'il faut, il fait son travail !*

## 5. Compteur Avancé avec Historique

Ajoutons un historique des modifications et des styles dynamiques !

### Version Finale Complète

```
import { useState } from 'react';
import './App.css';
import BoutonCompteur from './BoutonCompteur';

function App() {
  const [count, setCount] = useState(0);
  const [pas, setPas] = useState(1);
  const [historique, setHistorique] = useState([]);

  const augmenter = () => {
    const nouvelleValeur = count + pas;
    setCount(nouvelleValeur);
    ajouterHistorique(`+${pas} → ${nouvelleValeur}`);
  };

  const diminuer = () => {
    const nouvelleValeur = count - pas;
    setCount(nouvelleValeur);
    ajouterHistorique(`-${pas} → ${nouvelleValeur}`);
  };

  const reinitialiser = () => {
    setCount(0);
    ajouterHistorique('Réinitialisation → 0');
  };

  const ajouterHistorique = (action) => {
    const temps = new Date().toLocaleTimeString();
    setHistorique([...historique, `[${temps}] ${action}`]);
  };

  // Couleur dynamique selon la valeur
  const getCouleurDisplay = () => {
    if (count > 0) return '#27ae60';
    if (count < 0) return '#e74c3c';
    return 'white';
  };
```

```
return (
  <div className="App">
    <h1>🎯 Compteur Avancé</h1>

    <div
      className="compteur-display"
      style={{ borderColor: getCouleurDisplay() }}
    >
      <h2 style={{ color: getCouleurDisplay() }}>{count}</h2>
      <p style={{ fontSize: '1rem', marginTop: '10px' }}>
        {count > 0 ? '📈 Positif' : count < 0 ? '📉 Négatif' : '➡️ Neutre'}
      </p>
    </div>

    <div className="pas-selector">
      <label>Pas d'incrémentation : </label>
      <select value={pas} onChange={(e) => setPas(Number(e.target.value))}>
        <option value={1}>1</option>
        <option value={5}>5</option>
        <option value={10}>10</option>
        <option value={50}>50</option>
      </select>
    </div>

    <div className="boutons">
      <BoutonCompteur onClick={diminuer} couleur="#e74c3c">
        - {pas}
      </BoutonCompteur>

      <BoutonCompteur onClick={reinitialiser} couleur="#95a5a6">
        🔄 Reset
      </BoutonCompteur>

      <BoutonCompteur onClick={augmenter} couleur="#27ae60">
        + {pas}
      </BoutonCompteur>
    </div>

    {/* Historique */}
    <div className="historique">
      <h3>📜 Historique</h3>
      <div className="historique-liste">
```

```
          {historique.length === 0 ? (
            <p>Aucune action encore...</p>
          ) : (
            historique.slice(-5).reverse().map((action, index) => (
              <div key={index} className="historique-item">
                {action}
              </div>
            ))
          )}
        </div>
      </div>
    </div>
  );
}

export default App;
```

## Styles pour l'Historique

Ajoutez ces styles finaux dans App.css :

```
.historique {
  margin-top: 50px;
  background: rgba(255, 255, 255, 0.1);
  border-radius: 15px;
  padding: 25px;
  backdrop-filter: blur(10px);
  max-width: 500px;
}

.historique h3 {
  margin-top: 0;
  margin-bottom: 20px;
  font-size: 1.5rem;
}

.historique-liste {
  max-height: 200px;
  overflow-y: auto;
}

.historique-item {
  background: rgba(255, 255, 255, 0.2);
  padding: 10px 15px;
  margin: 8px 0;
  border-radius: 8px;
  font-size: 0.9rem;
  text-align: left;
}

/* Améliorer le compteur-display */
.compteur-display {
  border: 4px solid white;
  transition: all 0.3s ease;
}

.compteur-display h2 {
  transition: color 0.3s ease;
}
```

🎉 **Bravo !**

Vous avez créé une application de compteur complète avec :

- Gestion d'état avec useState
- Composants réutilisables avec props
- Styles dynamiques
- Historique des actions

## Fin de la Partie 1

*Dans la Partie 2, nous créerons une Todo App complète !*