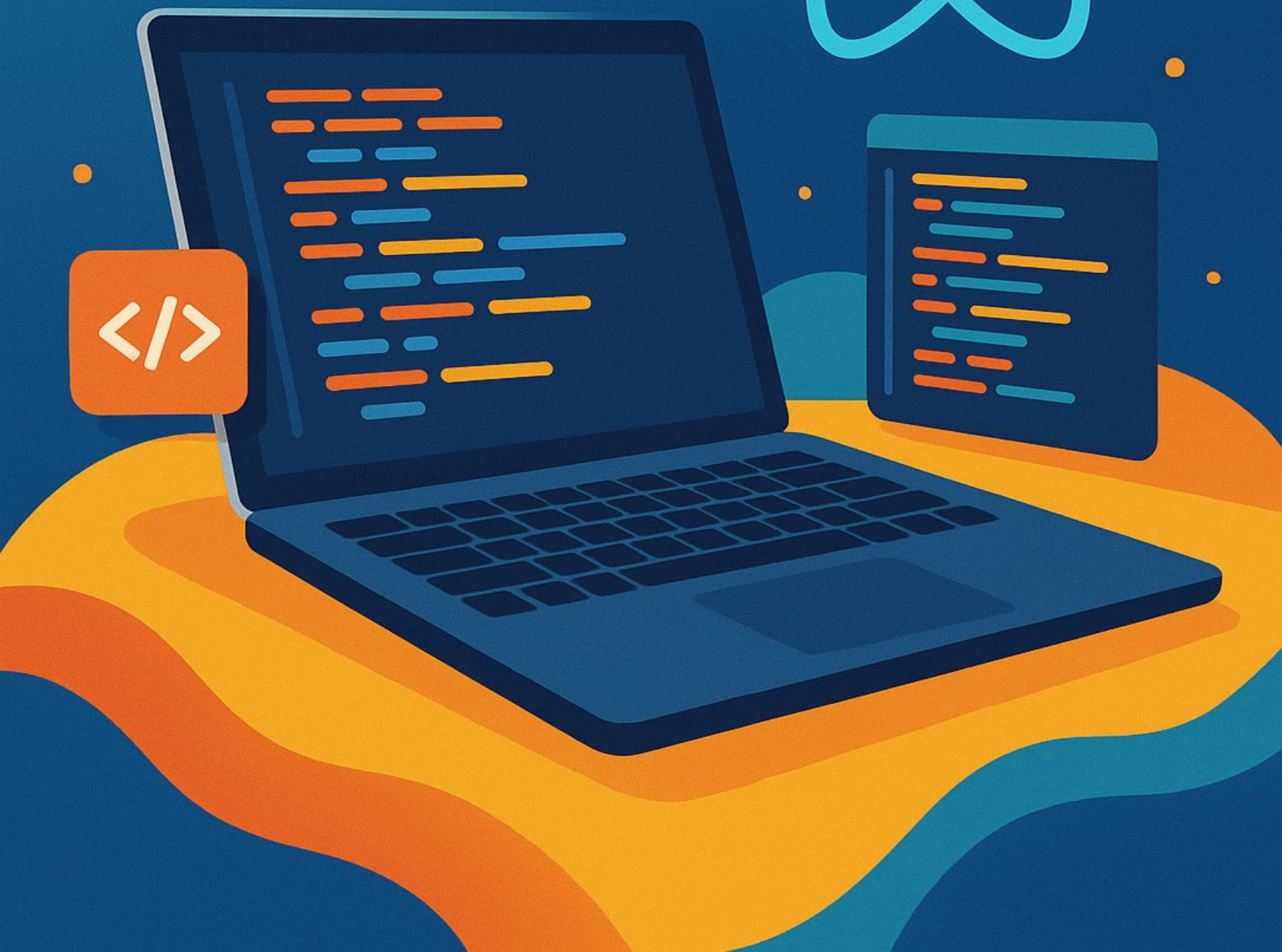
FORMATION REACT

NIVEAU INTERMÉDIAIRE



React JS Niveau Intermédiaire

Projets Concrets et Applications Réelles

📝 Créer une Todo App 📝

Partie 2 : Application Todo (Pages 16-25)

📝 Projet : Todo App Complète

Dans cette partie, nous allons créer une application de gestion de tâches professionnelle avec toutes les fonctionnalités essentielles !

Fonctionnalités

- ✅ Ajouter des tâches
- ✏️ Modifier des tâches
- 🗑️ Supprimer des tâches
- ✔️ Marquer comme complétée
- 🔍 Filtrer les tâches
- 📊 Statistiques en temps réel

1. Créer le Projet Todo

Commençons par créer un nouveau projet :

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

Structure de Base - App.js

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

function App() {
  const [taches, setTaches] = useState([]);
  const [inputValue, setInputValue] = useState('');

  return (
    <div className="App">
      <div className="container">
        <h1>📝 Ma Todo App</h1>

        {/* Formulaire d'ajout */}
        <div className="add-todo">
          <input
            type="text"
            value={inputValue}
            onChange={(e) => setInputValue(e.target.value)}
            placeholder="Ajouter une nouvelle tâche..."
          />
          <button>Ajouter</button>
        </div>

        {/* Liste des tâches */}
        <div className="todo-list">
          <p>Aucune tâche pour le moment</p>
        </div>
      </div>
    </div>
  );
}
```

```
export default App;
```

Styles - App.css

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

.App {
  min-height: 100vh;
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  padding: 50px 20px;
  font-family: 'Arial', sans-serif;
}

.container {
  max-width: 600px;
  margin: 0 auto;
  background: white;
  border-radius: 20px;
  padding: 30px;
  box-shadow: 0 20px 60px rgba(0, 0, 0, 0.3);
}

h1 {
  text-align: center;
  color: #667eea;
  margin-bottom: 30px;
  font-size: 2.5rem;
}

.add-todo {
  display: flex;
  gap: 10px;
  margin-bottom: 30px;
}

.add-todo input {
  flex: 1;
  padding: 15px;
  border: 2px solid #e0e0e0;
  border-radius: 10px;
  font-size: 1rem;
  transition: border-color 0.3s;
```

```
}

.add-todo input:focus {
  outline: none;
  border-color: #667eea;
}

.add-todo button {
  padding: 15px 30px;
  background: #667eea;
  color: white;
  border: none;
  border-radius: 10px;
  font-size: 1rem;
  font-weight: bold;
  cursor: pointer;
  transition: background 0.3s;
}

.add-todo button:hover {
  background: #5568d3;
}
```

2. Ajouter des Tâches

Implémentons la fonctionnalité d'ajout de tâches avec useState et les événements !

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

function App() {
  const [taches, setTaches] = useState([]);
  const [inputValue, setInputValue] = useState('');

  const ajouterTache = () => {
    if (inputValue.trim() === '') {
      alert('La tâche ne peut pas être vide !');
      return;
    }

    const nouvelleTache = {
      id: Date.now(),
      texte: inputValue,
      completee: false,
      dateCreation: new Date().toLocaleString()
    };

    setTaches([...taches, nouvelleTache]);
    setInputValue(''); // Vider l'input
  };

  const handleKeyPress = (e) => {
    if (e.key === 'Enter') {
      ajouterTache();
    }
  };

  return (
    <div className="App">
      <div className="container">
        <h1>📝 Ma Todo App</h1>

        <div className="add-todo">
          <input
            type="text"
            value={inputValue}
```

```
            onChange={(e) => setInputValue(e.target.value)}
            onKeyPress={handleKeyPress}
            placeholder="Ajouter une nouvelle tâche..."
          />
          <button onClick={ajouterTache}>Ajouter</button>
        </div>

        <div className="todo-list">
          {taches.length === 0 ? (
            <p className="empty">Aucune tâche pour le moment 🎉</p>
          ) : (
            taches.map((tache) => (
              <div key={tache.id} className="todo-item">
                <span>{tache.texte}</span>
              </div>
            ))
          )}
        </div>
      </div>
    </div>
  );
}

export default App;
```

💡 Date.now() pour l'ID ? C'est comme utiliser une horloge atomique pour savoir si vos œufs sont prêts !

Styles pour les Tâches

```
.todo-list {
  display: flex;
  flex-direction: column;
  gap: 10px;
}

.todo-item {
  display: flex;
  align-items: center;
  justify-content: space-between;
  padding: 15px;
  background: #f8f9fa;
  border-radius: 10px;
  transition: all 0.3s;
  border-left: 4px solid #667eea;
}

.todo-item:hover {
  background: #e9ecef;
  transform: translateX(5px);
}

.empty {
  text-align: center;
  color: #95a5a6;
  padding: 40px;
  font-size: 1.2rem;
}
```

✨ Essayez d'ajouter plusieurs tâches et appuyez sur Entrée pour ajouter rapidement !

3. Compléter et Supprimer

Ajoutons les fonctionnalités pour marquer une tâche comme complétée et la supprimer !

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

function App() {
  const [taches, setTaches] = useState([]);
  const [inputValue, setInputValue] = useState('');

  const ajouterTache = () => {
    if (inputValue.trim() === '') {
      alert('La tâche ne peut pas être vide !');
      return;
    }

    const nouvelleTache = {
      id: Date.now(),
      texte: inputValue,
      completee: false,
      dateCreation: new Date().toLocaleString()
    };

    setTaches([...taches, nouvelleTache]);
    setInputValue('');
  };

  const toggleTache = (id) => {
    setTaches(taches.map(tache =>
      tache.id === id
        ? { ...tache, completee: !tache.completee }
        : tache
    ));
  };

  const supprimerTache = (id) => {
    setTaches(taches.filter(tache => tache.id !== id));
  };

  const handleKeyPress = (e) => {
    if (e.key === 'Enter') {
```

```
      ajouterTache();
    }
  };

  return (
    <div className="App">
      <div className="container">
        <h1>📝 Ma Todo App</h1>

        <div className="add-todo">
          <input
            type="text"
            value={inputValue}
            onChange={(e) => setInputValue(e.target.value)}
            onKeyPress={handleKeyPress}
            placeholder="Ajouter une nouvelle tâche..."
          />
          <button onClick={ajouterTache}>Ajouter</button>
        </div>

        <div className="todo-list">
          {taches.length === 0 ? (
            <p className="empty">Aucune tâche pour le moment 🎉</p>
          ) : (
            taches.map((tache) => (
              <div
                key={tache.id}
                className={`todo-item ${tache.completee ? 'completed' : ''}`}
              >
                <div className="todo-content">
                  <input
                    type="checkbox"
                    checked={tache.completee}
                    onChange={() => toggleTache(tache.id)}
                  />
                  <span>{tache.texte}</span>
                </div>

                <button
                  className="btn-delete"
                  onClick={() => supprimerTache(tache.id)}
                >
                  🗑️
                </button>
```

```
              </div>
            ))
          )}
        </div>
      </div>
    </div>
  );
}

export default App;
```

Styles Mis à Jour

```
.todo-item {
  display: flex;
  align-items: center;
  justify-content: space-between;
  padding: 15px;
  background: #f8f9fa;
  border-radius: 10px;
  transition: all 0.3s;
  border-left: 4px solid #667eea;
}

.todo-item.completed {
  opacity: 0.6;
  border-left-color: #27ae60;
}

.todo-content {
  display: flex;
  align-items: center;
  gap: 15px;
  flex: 1;
}

.todo-content input[type="checkbox"] {
  width: 20px;
  height: 20px;
  cursor: pointer;
}

.todo-content span {
  font-size: 1.1rem;
}

.todo-item.completed span {
  text-decoration: line-through;
  color: #95a5a6;
}

.btn-delete {
  background: #e74c3c;
  color: white;
  border: none;
```

```
  padding: 8px 12px;
  border-radius: 5px;
  cursor: pointer;
  font-size: 1.2rem;
  transition: background 0.3s;
}

.btn-delete:hover {
  background: #c0392b;
}
```

> 💡 *Une checkbox qui barre du texte, c'est la satisfaction ultime du développeur ! Presque mieux qu'un café.*

4. Filtres et Statistiques

Ajoutons des filtres pour voir les tâches actives, complétées ou toutes, plus des statistiques !

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

function App() {
  const [taches, setTaches] = useState([]);
  const [inputValue, setInputValue] = useState('');
  const [filtre, setFiltre] = useState('toutes'); // 'toutes', 'actives',
'completees'

  const ajouterTache = () => {
    if (inputValue.trim() === '') return;

    setTaches([...taches, {
      id: Date.now(),
      texte: inputValue,
      completee: false,
      dateCreation: new Date().toLocaleString()
    }]);
    setInputValue('');
  };

  const toggleTache = (id) => {
    setTaches(taches.map(tache =>
      tache.id === id ? { ...tache, completee: !tache.completee } : tache
    ));
  };

  const supprimerTache = (id) => {
    setTaches(taches.filter(tache => tache.id !== id));
  };

  // Filtrer les tâches
  const tachesFiltrees = taches.filter(tache => {
    if (filtre === 'actives') return !tache.completee;
    if (filtre === 'completees') return tache.completee;
    return true;
  });
```

```
  // Statistiques
  const stats = {
    total: taches.length,
    actives: taches.filter(t => !t.completee).length,
    completees: taches.filter(t => t.completee).length
  };

  return (
    <div className="App">
      <div className="container">
        <h1>📝 Ma Todo App</h1>

        {/* Statistiques */}
        <div className="stats">
          <div className="stat-item">
            <span className="stat-number">{stats.total}</span>
            <span className="stat-label">Total</span>
          </div>
          <div className="stat-item">
            <span className="stat-number">{stats.actives}</span>
            <span className="stat-label">Actives</span>
          </div>
          <div className="stat-item">
            <span className="stat-number">{stats.completees}</span>
            <span className="stat-label">Complétées</span>
          </div>
        </div>

        <div className="add-todo">
          <input
            type="text"
            value={inputValue}
            onChange={(e) => setInputValue(e.target.value)}
            onKeyPress={(e) => e.key === 'Enter' && ajouterTache()}
            placeholder="Ajouter une nouvelle tâche..."
          />
          <button onClick={ajouterTache}>Ajouter</button>
        </div>

        {/* Filtres */}
        <div className="filtres">
          <button
            className={filtre === 'toutes' ? 'active' : ''}
            onClick={() => setFiltre('toutes')}
```

```
          >
            Toutes
          </button>
          <button
            className={filtre === 'actives' ? 'active' : ''}
            onClick={() => setFiltre('actives')}
          >
            Actives
          </button>
          <button
            className={filtre === 'completees' ? 'active' : ''}
            onClick={() => setFiltre('completees')}
          >
            Complétées
          </button>
        </div>

        <div className="todo-list">
          {tachesFiltrees.length === 0 ? (
            <p className="empty">Aucune tâche {filtre} 🎉</p>
          ) : (
            tachesFiltrees.map((tache) => (
              <div
                key={tache.id}
                className={`todo-item ${tache.completee ? 'completed' : ''}`}
              >
                <div className="todo-content">
                  <input
                    type="checkbox"
                    checked={tache.completee}
                    onChange={() => toggleTache(tache.id)}
                  />
                  <span>{tache.texte}</span>
                </div>

                <button
                  className="btn-delete"
                  onClick={() => supprimerTache(tache.id)}
                >
                  🗑️
                </button>
              </div>
            ))
          )}
```

```
        </div>
      </div>
    </div>
  );
}

export default App;
```

Styles Finaux Complets

```
.stats {
  display: flex;
  justify-content: space-around;
  margin-bottom: 30px;
  gap: 15px;
}

.stat-item {
  flex: 1;
  text-align: center;
  padding: 20px;
  background: linear-gradient(135deg, #667eea, #764ba2);
  border-radius: 10px;
  color: white;
}

.stat-number {
  display: block;
  font-size: 2.5rem;
  font-weight: bold;
  margin-bottom: 5px;
}

.stat-label {
  font-size: 0.9rem;
  opacity: 0.9;
}

.filtres {
  display: flex;
  gap: 10px;
  margin-bottom: 20px;
  justify-content: center;
}

.filtres button {
  padding: 10px 20px;
  background: #e0e0e0;
  color: #333;
  border: none;
  border-radius: 8px;
  cursor: pointer;
```

```
  font-weight: bold;
  transition: all 0.3s;
}

.filtres button:hover {
  background: #d0d0d0;
}

.filtres button.active {
  background: #667eea;
  color: white;
}
```

5. Version Finale avec Édition

Ajoutons la possibilité d'éditer une tâche en double-cliquant dessus !

Créez un nouveau composant **TodoItem.js** :

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

function TodoItem({ tache, toggleTache, supprimerTache, modifierTache }) {
  const [edition, setEdition] = useState(false);
  const [texteEdition, setTexteEdition] = useState(tache.texte);

  const sauvegarder = () => {
    if (texteEdition.trim()) {
      modifierTache(tache.id, texteEdition);
      setEdition(false);
    }
  };

  return (
    <div className={`todo-item ${tache.completee ? 'completed' : ''}`}>
      <div className="todo-content">
        <input
          type="checkbox"
          checked={tache.completee}
          onChange={() => toggleTache(tache.id)}
        />

        {edition ? (
          <input
            type="text"
            value={texteEdition}
            onChange={(e) => setTexteEdition(e.target.value)}
            onBlur={sauvegarder}
            onKeyPress={(e) => e.key === 'Enter' && sauvegarder()}
            className="edit-input"
            autoFocus
          />
        ) : (
          <span onDoubleClick={() => setEdition(true)}>
            {tache.texte}
          </span>
        )}
      </div>
```

```
      <div className="todo-actions">
        <button
          className="btn-edit"
          onClick={() => setEdition(true)}
        >
          ✏️
        </button>
        <button
          className="btn-delete"
          onClick={() => supprimerTache(tache.id)}
        >
          🗑️
        </button>
      </div>
    </div>
  );
}

export default TodoItem;
```

App.js Final

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

function App() {
  const [taches, setTaches] = useState([]);
  const [inputValue, setInputValue] = useState('');
  const [filtre, setFiltre] = useState('toutes');

  const ajouterTache = () => {
    if (inputValue.trim() === '') return;

    setTaches([...taches, {
      id: Date.now(),
      texte: inputValue,
      completee: false
    }]);
    setInputValue('');
  };

  const toggleTache = (id) => {
    setTaches(taches.map(t =>
      t.id === id ? { ...t, completee: !t.completee } : t
    ));
  };

  const supprimerTache = (id) => {
    setTaches(taches.filter(t => t.id !== id));
  };

  const modifierTache = (id, nouveauTexte) => {
    setTaches(taches.map(t =>
      t.id === id ? { ...t, texte: nouveauTexte } : t
    ));
  };

  const tachesFiltrees = taches.filter(t => {
    if (filtre === 'actives') return !t.completee;
    if (filtre === 'completees') return t.completee;
    return true;
  });
```

```
  const stats = {
    total: taches.length,
    actives: taches.filter(t => !t.completee).length,
    completees: taches.filter(t => t.completee).length
  };

  return (
    <div className="App">
      <div className="container">
        <h1>📝 Ma Todo App</h1>

        <div className="stats">
          <div className="stat-item">
            <span className="stat-number">{stats.total}</span>
            <span className="stat-label">Total</span>
          </div>
          <div className="stat-item">
            <span className="stat-number">{stats.actives}</span>
            <span className="stat-label">Actives</span>
          </div>
          <div className="stat-item">
            <span className="stat-number">{stats.completees}</span>
            <span className="stat-label">Complétées</span>
          </div>
        </div>

        <div className="add-todo">
          <input
            type="text"
            value={inputValue}
            onChange={(e) => setInputValue(e.target.value)}
            onKeyPress={(e) => e.key === 'Enter' && ajouterTache()}
            placeholder="Ajouter une nouvelle tâche..."
          />
          <button onClick={ajouterTache}>Ajouter</button>
        </div>

        <div className="filtres">
          <button
            className={filtre === 'toutes' ? 'active' : ''}
            onClick={() => setFiltre('toutes')}
          >
            Toutes
          </button>
```

```
          <button
            className={filtre === 'actives' ? 'active' : ''}
            onClick={() => setFiltre('actives')}
          >
            Actives
          </button>
          <button
            className={filtre === 'completees' ? 'active' : ''}
            onClick={() => setFiltre('completees')}
          >
            Complétées
          </button>
        </div>

        <div className="todo-list">
          {tachesFiltrees.length === 0 ? (
            <p className="empty">Aucune tâche {filtre} 🎉</p>
          ) : (
            tachesFiltrees.map((tache) => (
              <TodoItem
                key={tache.id}
                tache={tache}
                toggleTache={toggleTache}
                supprimerTache={supprimerTache}
                modifierTache={modifierTache}
              />
            ))
          )}
        </div>
      </div>
    </div>
  );
}

export default App;
```

Styles Supplémentaires

```
.todo-actions {
  display: flex;
  gap: 8px;
}

.btn-edit {
  background: #3498db;
  color: white;
  border: none;
  padding: 8px 12px;
  border-radius: 5px;
  cursor: pointer;
  font-size: 1.2rem;
  transition: background 0.3s;
}

.btn-edit:hover {
  background: #2980b9;
}

.edit-input {
  flex: 1;
  padding: 5px 10px;
  border: 2px solid #667eea;
  border-radius: 5px;
  font-size: 1.1rem;
}
```

🎉 Félicitations !

Vous avez créé une Todo App complète et professionnelle ! Vous maîtrisez maintenant :

- useState pour gérer l'état complexe
- Props pour passer des données et fonctions
- Événements et formulaires
- Rendu conditionnel et listes
- Composants réutilisables

💡 Avec cette Todo App, vous pourriez même gérer votre vie... ou au moins essayer !

🚀 Continuez à pratiquer et à créer ! 🚀

Fin de l'Ebook Niveau Intermédiaire