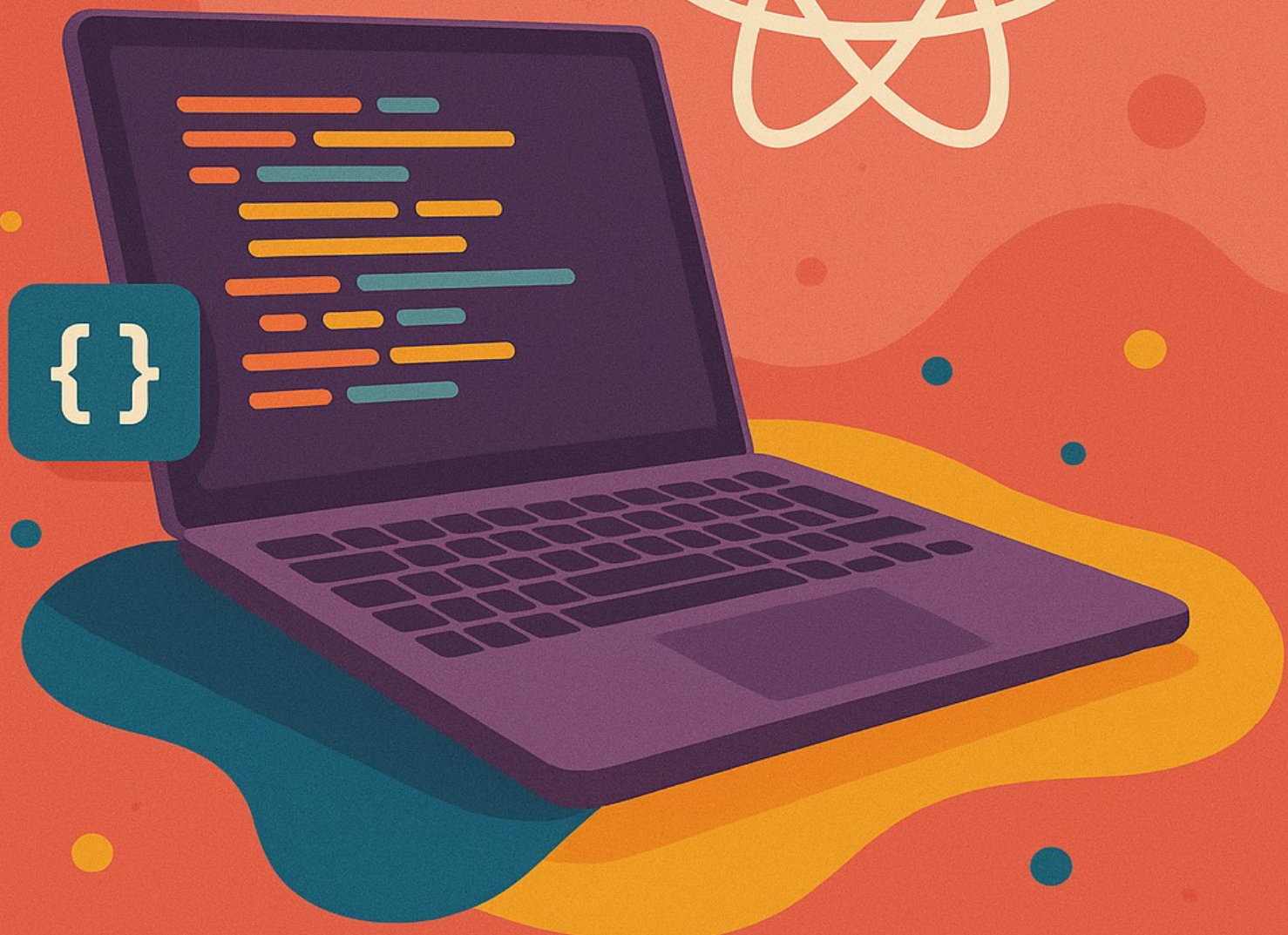
EXERCICES ET CORRIGÉS REACT JS



React JS

10 Exercices Corrigés

Du Niveau Débutant au Niveau Avancé

Partie 1 : Exercices 1-3 (Niveau Débutant)

📚 Introduction

Bienvenue dans cette série d'exercices pratiques React ! Chaque exercice est conçu pour renforcer vos compétences en programmation React, du niveau débutant au niveau avancé.

Comment utiliser cet ebook

- Lisez attentivement l'énoncé de l'exercice
- Essayez de résoudre l'exercice par vous-même
- Consultez les indices si vous êtes bloqué
- Étudiez la solution détaillée et les explications

Prérequis

Pour ces exercices, vous aurez besoin de :

- Node.js installé (version 14 ou supérieure)
- Un éditeur de code (VS Code recommandé)
- Connaissances de base en JavaScript

Exercice 1 : Carte de Profil

⭐ **Niveau : Débutant**

📋 Énoncé

> 📝 **Créez un composant React qui affiche une carte de profil avec un nom, un âge, une profession et une photo. Le composant doit recevoir ces informations via les props.**

🎯 Objectifs d'apprentissage

- Créer un composant fonctionnel
- Utiliser les props pour passer des données
- Appliquer du style inline en JSX
- Afficher des images dans React

📝 Spécifications

Votre composant doit :

- S'appeler **CarteProfi**
- Accepter les props : nom, age, profession, photo
- Avoir une bordure et un style attrayant
- Afficher une photo circulaire

> 💡 *Conseil : Utilisez des styles inline avec l'objet style={{}} pour styliser votre composant.*

💭 Indices

Si vous êtes bloqué, voici quelques indices :

- **Indice 1 :** Utilisez la destructuration des props : function CarteProfil({ nom, age, profession, photo })
- **Indice 2 :** Pour une image ronde, utilisez borderRadius: '50%'
- **Indice 3 :** Les props se passent comme des attributs HTML : <CarteProfil nom='Marie' />

✅ Solution Complète

Fichier : CarteProfil.js

```
function CarteProfil({ nom, age, profession, photo }) {
  return (
    <div style={{
      border: '2px solid #3498db',
      borderRadius: '15px',
      padding: '25px',
      margin: '20px',
      maxWidth: '300px',
      textAlign: 'center',
      boxShadow: '0 4px 8px rgba(0, 0, 0, 0.1)',
      backgroundColor: '#fff'
    }}>
      {/* Photo de profil */}
      <img
        src={photo}
        alt={nom}
        style={{
          width: '120px',
          height: '120px',
          borderRadius: '50%',
          objectFit: 'cover',
          marginBottom: '15px',
          border: '3px solid #3498db'
        }}
      />

      {/* Informations */}
      <h2 style={{
```

```
        color: '#2c3e50',
        marginBottom: '10px'
      }}>
        {nom}
      </h2>

      <p style={{
        color: '#7f8c8d',
        fontSize: '1.1rem',
        marginBottom: '5px'
      }}>
        {age} ans
      </p>

      <p style={{
        color: '#3498db',
        fontSize: '1rem',
        fontWeight: 'bold'
      }}>
        {profession}
      </p>
    </div>
  );
}

export default CarteProfil;
```

Fichier : App.js (Utilisation)

```
import CarteProfil from './CarteProfil';

function App() {
  return (
    <div style={{
      display: 'flex',
      flexWrap: 'wrap',
      justifyContent: 'center',
      padding: '20px',
      backgroundColor: '#ecf0f1',
      minHeight: '100vh'
    }}>
      <CarteProfil
        nom="Marie Dupont"
        age={28}
        profession="Développeuse Web"
        photo="https://i.pravatar.cc/150?img=1"
      />

      <CarteProfil
        nom="Jean Martin"
        age={32}
        profession="Designer UX/UI"
        photo="https://i.pravatar.cc/150?img=12"
      />

      <CarteProfil
        nom="Sophie Bernard"
        age={25}
        profession="Chef de Projet"
        photo="https://i.pravatar.cc/150?img=5"
      />
    </div>
  );
}

export default App;
```

📖 Explications Détaillées

1. Destructuration des props

Au lieu d'écrire `props.nom`, nous utilisons la destructuration : `{ nom, age, profession, photo }`. C'est plus propre et plus lisible.

2. Styles inline

En React, les styles inline utilisent des objets JavaScript. Les propriétés CSS en kebab-case (border-radius) deviennent camelCase (borderRadius). Les valeurs sont entre guillemets.

3. Image ronde

`borderRadius: '50%'` transforme une image carrée en cercle. `objectFit: 'cover'` assure que l'image remplit bien le cercle sans déformation.

> **✅ Résultat : Vous avez créé un composant réutilisable qui peut afficher différents profils !**

Exercice 2 : Compteur Simple

⭐⭐ **Niveau : Débutant+**

📋 Énoncé

📝 **Créez un compteur avec trois boutons : un pour augmenter (+1), un pour diminuer (-1), et un pour réinitialiser à zéro. Le compteur doit afficher la valeur actuelle et changer de couleur selon qu'elle est positive, négative ou nulle.**

🎯 Objectifs d'apprentissage

- Utiliser le hook useState
- Gérer des événements (onClick)
- Appliquer des styles conditionnels
- Créer des fonctions de gestion d'événements

📝 Spécifications

- Valeur initiale : 0
- Couleur verte si positif
- Couleur rouge si négatif
- Couleur grise si zéro

💡 *Conseil : Créez une fonction qui retourne la couleur appropriée selon la valeur du compteur.*

💭 Indices

- **Indice 1 :** useState(0) initialise le state à 0
- **Indice 2 :** Pour incrémenter : setCount(count + 1)
- **Indice 3 :** Utilisez if/else ou l'opérateur ternaire pour la couleur

✅ Solution Complète

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

function Compteur() {
  // État du compteur
  const [count, setCount] = useState(0);

  // Fonctions de gestion
  const augmenter = () => {
    setCount(count + 1);
  };

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

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

  // Fonction pour déterminer la couleur
  const getCouleur = () => {
    if (count > 0) return '#27ae60'; // Vert
    if (count < 0) return '#e74c3c'; // Rouge
    return '#95a5a6'; // Gris
  };

  return (
    <div style={{
      display: 'flex',
      flexDirection: 'column',
      alignItems: 'center',
      justifyContent: 'center',
      minHeight: '100vh',
      backgroundColor: '#ecf0f1',
```

```
      fontFamily: 'Arial, sans-serif'
    }}>
      <h1 style={{ color: '#2c3e50', marginBottom: '30px' }}>
        Compteur Simple
      </h1>

      {/* Affichage du compteur */}
      <div style={{
        fontSize: '5rem',
        fontWeight: 'bold',
        color: getCouleur(),
        marginBottom: '30px',
        padding: '30px 60px',
        backgroundColor: 'white',
        borderRadius: '15px',
        boxShadow: '0 4px 6px rgba(0, 0, 0, 0.1)',
        minWidth: '200px',
        textAlign: 'center'
      }}>
        {count}
      </div>

      {/* Indicateur de statut */}
      <p style={{
        fontSize: '1.5rem',
        color: getCouleur(),
        marginBottom: '30px',
        fontWeight: 'bold'
      }}>
        {count > 0 && '📈 Positif'}
        {count < 0 && '📉 Négatif'}
        {count === 0 && '➡️ Neutre'}
      </p>

      {/* Boutons */}
      <div style={{ display: 'flex', gap: '15px' }}>
        <button
          onClick={diminuer}
          style={{
            padding: '15px 30px',
            fontSize: '1.2rem',
            fontWeight: 'bold',
            backgroundColor: '#e74c3c',
            color: 'white',
```

```
            border: 'none',
            borderRadius: '10px',
            cursor: 'pointer',
            transition: 'transform 0.2s'
          }}
          onMouseEnter={(e) => e.target.style.transform = 'scale(1.05)'}
          onMouseLeave={(e) => e.target.style.transform = 'scale(1)'}
        >
          - Diminuer
        </button>

        <button
          onClick={reinitialiser}
          style={{
            padding: '15px 30px',
            fontSize: '1.2rem',
            fontWeight: 'bold',
            backgroundColor: '#95a5a6',
            color: 'white',
            border: 'none',
            borderRadius: '10px',
            cursor: 'pointer'
          }}
        >
          🔄 Reset
        </button>

        <button
          onClick={augmenter}
          style={{
            padding: '15px 30px',
            fontSize: '1.2rem',
            fontWeight: 'bold',
            backgroundColor: '#27ae60',
            color: 'white',
            border: 'none',
            borderRadius: '10px',
            cursor: 'pointer'
          }}
        >
          + Augmenter
        </button>
      </div>
    </div>
```

```
  );
}

export default Compteur;
```

📖 Explications Détaillées

1. useState Hook

```
const [count, setCount] = useState(0);
```

count : la valeur actuelle du compteur

setCount : la fonction pour modifier count

useState(0) : initialise count à 0

2. Fonctions de gestion

Nous créons des fonctions séparées (augmenter, diminuer, reinitialiser) plutôt que d'écrire la logique directement dans onClick. C'est plus propre et plus maintenable.

3. Couleur conditionnelle

```
const getCouleur = () => {
  if (count > 0) return '#27ae60';
  if (count < 0) return '#e74c3c';
  return '#95a5a6';
};
```

Cette fonction retourne une couleur différente selon la valeur du compteur. Elle est appelée dans le style pour changer dynamiquement la couleur.

4. Événements de souris

```
onMouseEnter={(e) => e.target.style.transform = 'scale(1.05)'}
onMouseLeave={(e) => e.target.style.transform = 'scale(1)'}
```

Ces événements ajoutent un effet d'agrandissement au survol du bouton, rendant l'interface plus interactive.

> **✅ Résultat : Un compteur fonctionnel avec des styles dynamiques et une interface attractive !**

Exercice 3 : Liste de Courses

⭐⭐ **Niveau : Débutant+**

📋 Énoncé

📝 **Créez une application de liste de courses où l'utilisateur peut ajouter des articles via un input, les afficher dans une liste, et les supprimer individuellement. Affichez aussi le nombre total d'articles.**

🎯 Objectifs d'apprentissage

- Gérer un tableau dans le state
- Utiliser .map() pour afficher une liste
- Gérer les formulaires (input)
- Ajouter et supprimer des éléments d'un tableau
- Utiliser les keys dans les listes

📝 Spécifications

- Un champ input pour saisir un article
- Un bouton pour ajouter l'article
- Une liste affichant tous les articles
- Un bouton de suppression pour chaque article
- Affichage du nombre total d'articles

💡 *Conseil : Utilisez Date.now() pour générer des IDs uniques pour chaque article.*

💭 Indices

- **Indice 1 :** Utilisez deux states : un pour l'input, un pour la liste
- **Indice 2 :** Pour ajouter : setListe([...liste, nouvelArticle])
- **Indice 3 :** Pour supprimer : setListe(liste.filter(item => item.id !== id))

✅ Solution Complète

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

function ListeCourses() {
  const [articles, setArticles] = useState([]);
  const [inputValue, setInputValue] = useState('');

  // Ajouter un article
  const ajouterArticle = () => {
    if (inputValue.trim() === '') {
      alert('Veuillez entrer un article !');
      return;
    }

    const nouvelArticle = {
      id: Date.now(),
      nom: inputValue
    };

    setArticles([...articles, nouvelArticle]);
    setInputValue(''); // Vider l'input
  };

  // Supprimer un article
  const supprimerArticle = (id) => {
    setArticles(articles.filter(article => article.id !== id));
  };

  // Gérer la touche Entrée
  const handleKeyPress = (e) => {
    if (e.key === 'Enter') {
      ajouterArticle();
    }
  };
```

```
  return (
    <div style={{
      maxWidth: '500px',
      margin: '50px auto',
      padding: '30px',
      backgroundColor: 'white',
      borderRadius: '15px',
      boxShadow: '0 4px 10px rgba(0, 0, 0, 0.1)',
      fontFamily: 'Arial, sans-serif'
    }}>
      <h1 style={{
        textAlign: 'center',
        color: '#2c3e50',
        marginBottom: '30px'
      }}>
        🛒 Ma Liste de Courses
      </h1>

      {/* Compteur d'articles */}
      <div style={{
        textAlign: 'center',
        padding: '15px',
        backgroundColor: '#3498db',
        color: 'white',
        borderRadius: '10px',
        marginBottom: '20px',
        fontSize: '1.2rem',
        fontWeight: 'bold'
      }}>
        {articles.length} article{articles.length !== 1 ? 's' : ''}
      </div>

      {/* Zone d'ajout */}
      <div style={{
        display: 'flex',
        gap: '10px',
        marginBottom: '25px'
      }}>
        <input
          type="text"
          value={inputValue}
          onChange={(e) => setInputValue(e.target.value)}
          onKeyPress={handleKeyPress}
          placeholder="Ajouter un article..."
```

```
          style={{
            flex: 1,
            padding: '12px',
            border: '2px solid #e0e0e0',
            borderRadius: '8px',
            fontSize: '1rem',
            outline: 'none'
          }}
        />
        <button
          onClick={ajouterArticle}
          style={{
            padding: '12px 25px',
            backgroundColor: '#27ae60',
            color: 'white',
            border: 'none',
            borderRadius: '8px',
            fontSize: '1rem',
            fontWeight: 'bold',
            cursor: 'pointer'
          }}
        >
          ➕ Ajouter
        </button>
      </div>

      {/* Liste des articles */}
      {articles.length === 0 ? (
        <p style={{
          textAlign: 'center',
          color: '#95a5a6',
          padding: '30px',
          fontSize: '1.1rem'
        }}>
          Votre liste est vide. Ajoutez votre premier article ! 🎉
        </p>
      ) : (
        <ul style={{
          listStyle: 'none',
          padding: 0,
          margin: 0
        }}>
          {articles.map((article) => (
            <li
```

```
              key={article.id}
              style={{
                display: 'flex',
                justifyContent: 'space-between',
                alignItems: 'center',
                padding: '15px',
                marginBottom: '10px',
                backgroundColor: '#f8f9fa',
                borderRadius: '8px',
                borderLeft: '4px solid #3498db'
              }}
            >
              <span style={{
                fontSize: '1.1rem',
                color: '#2c3e50'
              }}>
                {article.nom}
              </span>

              <button
                onClick={() => supprimerArticle(article.id)}
                style={{
                  padding: '8px 15px',
                  backgroundColor: '#e74c3c',
                  color: 'white',
                  border: 'none',
                  borderRadius: '5px',
                  cursor: 'pointer',
                  fontSize: '0.9rem'
                }}
              >
                🗑️ Supprimer
              </button>
            </li>
          ))}
        </ul>
      )}
    </div>
  );
}

export default ListeCourses;
```

📖 Explications Détaillées

1. Deux states différents

```
const [articles, setArticles] = useState([]);
const [inputValue, setInputValue] = useState('');
```

articles : tableau contenant tous les articles

inputValue : valeur actuelle de l'input

2. Ajouter un article

```
const nouvelArticle = {
  id: Date.now(),
  nom: inputValue
};
setArticles([...articles, nouvelArticle]);
```

...articles : spread operator qui copie tous les articles existants

Date.now() : génère un ID unique basé sur le timestamp

On ajoute le nouvel article à la fin du tableau

3. Supprimer un article

```
setArticles(articles.filter(article => article.id !== id));
```

La méthode **filter()** crée un nouveau tableau contenant uniquement les articles dont l'ID est différent de celui à supprimer.

4. Afficher la liste avec map()

```
{articles.map((article) => (
  <li key={article.id}>
    {article.nom}
  </li>
))}
```

map() transforme chaque élément du tableau en JSX

key={article.id} est obligatoire pour aider React à identifier chaque élément

> **✅ Résultat : Une application fonctionnelle de liste de courses avec ajout et suppression d'articles !**

🎉 Félicitations !

Vous avez terminé les 3 premiers exercices ! Vous maîtrisez maintenant :

- Les composants et les props
- Le hook useState
- La gestion d'événements
- Les listes et le rendu dynamique
- Les styles conditionnels

Fin de la Partie 1

Dans la Partie 2, nous passerons au niveau intermédiaire !