Exercice 3 - Liste de courses

Objectif

Afficher une liste dynamique à partir d'un tableau JavaScript.

Concepts abordés

- Méthode `.map()` pour transformer un tableau
- Prop `key` obligatoire
- Rendu de listes en JSX

Contexte

Vous créez une application de liste de courses. Pour l'instant, la liste est statique (pas d'ajout/suppression).

Énoncé

Étape 1 : Liste simple

Créer un composant `ListeCourses` qui affiche une liste d'articles à partir d'un tableau de strings :

```
const articles = ["Pain", "Lait", "Oeufs", "Beurre", "Farine"];
```

Étape 2 : Liste avec objets

Modifier pour utiliser un tableau d'objets avec id, nom et quantité :

```
const articles = [
  { id: 1, nom: "Pain", quantite: 2 },
  { id: 2, nom: "Lait", quantite: 1 },
  { id: 3, nom: "Oeufs", quantite: 12 },
  { id: 4, nom: "Beurre", quantite: 1 },
  { id: 5, nom: "Farine", quantite: 1 },
];
```

Étape 3 : Afficher le nombre total d'articles

Calculer et afficher le nombre total d'articles dans la liste.

Résultat attendu

```
┌──────────────────────────────────┐
│  Ma liste de courses             │
```

```
┌──────────────────────────────┐
│                              │
│  • Pain (x2)                 │
│  • Lait (x1)                 │
│  • Oeufs (x12)               │
│  • Beurre (x1)               │
│  • Farine (x1)               │
│                              │
│  Total : 5 articles          │
└──────────────────────────────┘
```

Indices

▶ Indice 1 : Syntaxe de base de map()

```
{tableau.map((element) => (
  <li>{element}</li>
))}
```

▶ Indice 2 : Utiliser la key avec un objet

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

▶ Indice 3 : Calculer le total

```
const total = articles.length;
// ou pour la somme des quantités :
const totalQuantites = articles.reduce((acc, art) => acc + art.quantite, 0);
```

Points d'attention

La prop key est OBLIGATOIRE

```
// ERREUR : React affiche un warning
{articles.map((article) => (
  <li>{article.nom}</li>
))}

// CORRECT : chaque élément a une key unique
```

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

Ne pas utiliser l'index comme key (sauf cas particuliers)

```
// À ÉVITER : si la liste change, les index changent aussi
{articles.map((article, index) => (
  <li key={index}>{article.nom}</li>
))}

// PRÉFÉRER : un identifiant stable
{articles.map((article) => (
  <li key={article.id}>{article.nom}</li>
))}
```

Quand l'index est acceptable

- Liste statique (jamais modifiée)
- Pas de réordonnancement
- Pas de suppression/ajout au milieu

La key doit être sur l'élément racine du map

```
// ERREUR : key sur un élément enfant
{articles.map((article) => (
  <li>
    <span key={article.id}>{article.nom}</span>
  </li>
))}

// CORRECT : key sur l'élément retourné par map
{articles.map((article) => (
  <li key={article.id}>
    <span>{article.nom}</span>
  </li>
))}
```

Correction

```
// src/exercices/ex03-liste-courses/ListeCourses.jsx

function ListeCourses() {
  // ═══════════════════════════════════════════════════════════
  // DONNÉES
```

```
    // Dans une vraie app, ces données viendraient d'une API ou du state
    // Ici, elles sont "en dur" pour l'exercice
    // ═══════════════════════════════════════════════════════════════
    const articles = [
      { id: 1, nom: "Pain", quantite: 2 },
      { id: 2, nom: "Lait", quantite: 1 },
      { id: 3, nom: "Oeufs", quantite: 12 },
      { id: 4, nom: "Beurre", quantite: 1 },
      { id: 5, nom: "Farine", quantite: 1 },
    ];

    // ═══════════════════════════════════════════════════════════════
    // CALCUL DU TOTAL
    // articles.length = nombre d'éléments différents (5)
    // reduce = somme de toutes les quantités (17)
    // ═══════════════════════════════════════════════════════════════
    const nombreArticles = articles.length;
    const totalQuantites = articles.reduce(
      (accumulator, article) => accumulator + article.quantite,
      0  // Valeur initiale de l'accumulator
    );

    return (
      <div className="liste-courses">
        <h2>Ma liste de courses</h2>

        {/* ═══════════════════════════════════════════════════════
            AFFICHAGE DE LA LISTE AVEC MAP()

            .map() transforme chaque élément du tableau en JSX
            C'est comme une boucle for, mais qui retourne un nouveau tableau

            articles.map(article => ...) est équivalent à :

            const resultat = [];
            for (const article of articles) {
              resultat.push(<li>...</li>);
            }
            return resultat;

            La KEY est obligatoire pour que React puisse identifier
            chaque élément de manière unique lors des mises à jour.
        ═══════════════════════════════════════════════════════ */}
        <ul>
          {articles.map((article) => (
            <li key={article.id}>
              {article.nom} (x{article.quantite})
            </li>
          ))}
        </ul>

        {/* ═══════════════════════════════════════════════════════
            AFFICHAGE DES TOTAUX
        ═══════════════════════════════════════════════════════ */}
```

```
      <p>
        <strong>Total : </strong>
        {nombreArticles} articles différents ({totalQuantites} au total)
      </p>
    </div>
  );
}

export default ListeCourses;
```

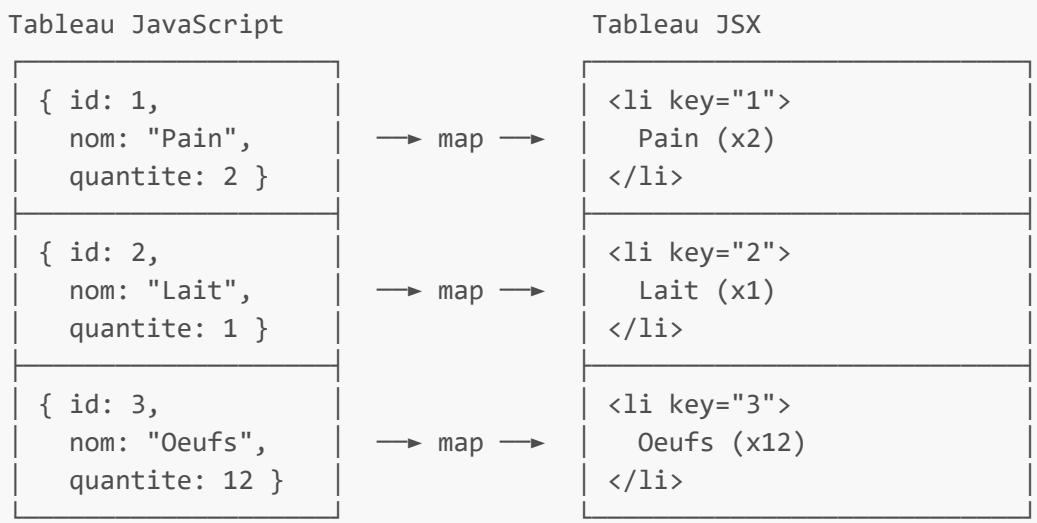
Version alternative : Composant Article séparé

```
// Article.jsx - Composant pour un seul article
function Article({ nom, quantite }) {
  return (
    <li className="article">
      <span className="nom">{nom}</span>
      <span className="quantite">x{quantite}</span>
    </li>
  );
}

export default Article;
```

```
// ListeCourses.jsx - Utilise le composant Article
import Article from './Article';

function ListeCourses() {
  const articles = [
    { id: 1, nom: "Pain", quantite: 2 },
    { id: 2, nom: "Lait", quantite: 1 },
    // ...
  ];

  return (
    <div className="liste-courses">
      <h2>Ma liste de courses</h2>

      <ul>
        {articles.map((article) => (
          // La key reste sur l'élément retourné par map
          // Les autres données sont passées en props
          <Article
            key={article.id}
            nom={article.nom}
            quantite={article.quantite}
          />
        ))}
      </ul>
```

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

Schéma mental : Comment map() fonctionne



Pour aller plus loin

1. Ajouter une classe CSS différente si la quantité est > 5
2. Afficher "Liste vide" si le tableau est vide
3. Filtrer pour n'afficher que les articles avec quantité > 1

```jsx
// Classe conditionnelle
<li className={article.quantite > 5 ? "beaucoup" : ""}>

// Gérer la liste vide
{articles.length === 0 ? (
  <p>Liste vide</p>
) : (
  <ul>...</ul>
)}

// Filtrer avant le map
{articles
  .filter(article => article.quantite > 1)
  .map(article => (
    <li key={article.id}>{article.nom}</li>
  ))
}
```