Exercice 5 - Carte utilisateur avec Props

Objectif

Créer un composant réutilisable qui reçoit ses données via les props.

Concepts abordés

- Props : passage de données parent → enfant
- Déstructuration des props
- Props par défaut
- Typage implicite des props

Contexte

Vous créez un composant `UserCard` réutilisable pour afficher les informations d'un utilisateur. Ce composant sera utilisé à plusieurs endroits de l'application avec des données différentes.

Énoncé

Étape 1 : Composant de base

Créer un composant `UserCard` qui reçoit et affiche :

- `name` : le nom de l'utilisateur
- `email` : son email
- `role` : son rôle (Admin, User, etc.)

Étape 2 : Ajouter des props optionnelles

- `avatar` : URL de l'image (optionnel, afficher une image par défaut si absent)
- `isOnline` : booléen pour afficher un indicateur de statut

Étape 3 : Utiliser le composant

Dans un composant parent `UserList`, afficher 3 utilisateurs différents en utilisant `UserCard`.

Résultat attendu

Indices

▶ Indice 1 : Recevoir les props

```
// Avec déstructuration (recommandé)
function UserCard({ name, email, role }) {
  return <div>{name}</div>;
}

// Sans déstructuration
function UserCard(props) {
  return <div>{props.name}</div>;
}
```

▶ Indice 2 : Props par défaut

```
function UserCard({ name, avatar = "/default-avatar.png", isOnline = false }) {
  // avatar aura la valeur par défaut si non fourni
}
```

▶ Indice 3 : Passer les props au composant

```
<UserCard
  name="Victor"
  email="victor@email.com"
  role="Admin"
  isOnline={true}
/>
```

Points d'attention

Déstructuration dans les paramètres

```
// RECOMMANDÉ : déstructuration directe
function UserCard({ name, email }) {
  return <p>{name} - {email}</p>;
}

// ACCEPTABLE : déstructuration dans le corps
function UserCard(props) {
  const { name, email } = props;
  return <p>{name} - {email}</p>;
}
```

```
// FONCTIONNE mais verbeux
function UserCard(props) {
  return <p>{props.name} - {props.email}</p>;
}
```

Types de valeurs en props

```
// String : avec guillemets
<UserCard name="Victor" />

// Number : avec accolades
<UserCard age={26} />

// Boolean true : juste le nom ou avec accolades
<UserCard isOnline />
<UserCard isOnline={true} />

// Boolean false : obligatoirement avec accolades
<UserCard isOnline={false} />

// Objet : double accolades
<UserCard style={{ color: 'red' }} />
<UserCard user={{ name: 'Victor', age: 26 }} />
```

Props en lecture seule

```
function UserCard({ name }) {
  // INTERDIT : ne jamais modifier une prop
  name = "Autre";  // NON !

  // OK : créer une nouvelle variable
  const displayName = name.toUpperCase();
}
```

Correction

UserCard.jsx

```
// src/exercices/ex05-carte-utilisateur/UserCard.jsx

// ═══════════════════════════════════════════════════════════════
// COMPOSANT USERCARD
//
// Composant réutilisable pour afficher les informations d'un utilisateur.
// Il reçoit toutes ses données via les PROPS (propriétés).
```

```jsx
//
// Props :
//   - name (string, requis) : nom de l'utilisateur
//   - email (string, requis) : email de l'utilisateur
//   - role (string, requis) : rôle de l'utilisateur
//   - avatar (string, optionnel) : URL de l'image
//   - isOnline (boolean, optionnel) : statut de connexion
// ============================================================

function UserCard({
  name,
  email,
  role,
  avatar = "https://via.placeholder.com/80",  // Valeur par défaut
  isOnline = false                             // Valeur par défaut
}) {
  // ============================================================
  // LOGIQUE D'AFFICHAGE
  // On peut préparer des variables pour simplifier le JSX
  // ============================================================
  const statusClass = isOnline ? "status-online" : "status-offline";
  const statusText = isOnline ? "En ligne" : "Hors ligne";

  return (
    <article className="user-card">
      {/* En-tête avec avatar et statut */}
      <header className="user-card-header">
        <img
          src={avatar}
          alt={`Avatar de ${name}`}
          className="user-avatar"
        />
        <span className={`user-status ${statusClass}`}>
          {statusText}
        </span>
      </header>

      {/* Informations */}
      <div className="user-card-body">
        <h3 className="user-name">{name}</h3>
        <p className="user-email">{email}</p>
        <span className="user-role">{role}</span>
      </div>
    </article>
  );
}

export default UserCard;
```

UserList.jsx (Composant parent)

```jsx
// src/exercices/ex05-carte-utilisateur/UserList.jsx

import UserCard from './UserCard';

function UserList() {
  // ═══════════════════════════════════════════════════════════
  // DONNÉES DES UTILISATEURS
  // Dans une vraie app, ces données viendraient d'une API
  // ═══════════════════════════════════════════════════════════
  const users = [
    {
      id: 1,
      name: "Victor Besson",
      email: "victor@email.com",
      role: "Admin",
      avatar: "https://i.pravatar.cc/80?img=1",
      isOnline: true
    },
    {
      id: 2,
      name: "Alice Martin",
      email: "alice@email.com",
      role: "User",
      // Pas d'avatar → utilisera la valeur par défaut
      isOnline: false
    },
    {
      id: 3,
      name: "Bob Dupont",
      email: "bob@email.com",
      role: "Modérateur",
      avatar: "https://i.pravatar.cc/80?img=3",
      isOnline: true
    }
  ];

  return (
    <div className="user-list">
      <h2>Équipe</h2>

      <div className="user-list-grid">
        {/* ═══════════════════════════════════════════════════
            PASSAGE DES PROPS

            Méthode 1 : Props explicites (plus clair)
            <UserCard
              name={user.name}
              email={user.email}
              ...
            />

            Méthode 2 : Spread operator (plus concis)
            <UserCard {...user} />
```

```
            On utilise la méthode 1 pour la clarté pédagogique.
          ═══════════════════════════════════════════════════════ */}
          {users.map(user => (
            <UserCard
              key={user.id}
              name={user.name}
              email={user.email}
              role={user.role}
              avatar={user.avatar}
              isOnline={user.isOnline}
            />
          ))}
        </div>
      </div>
    );
}

export default UserList;
```

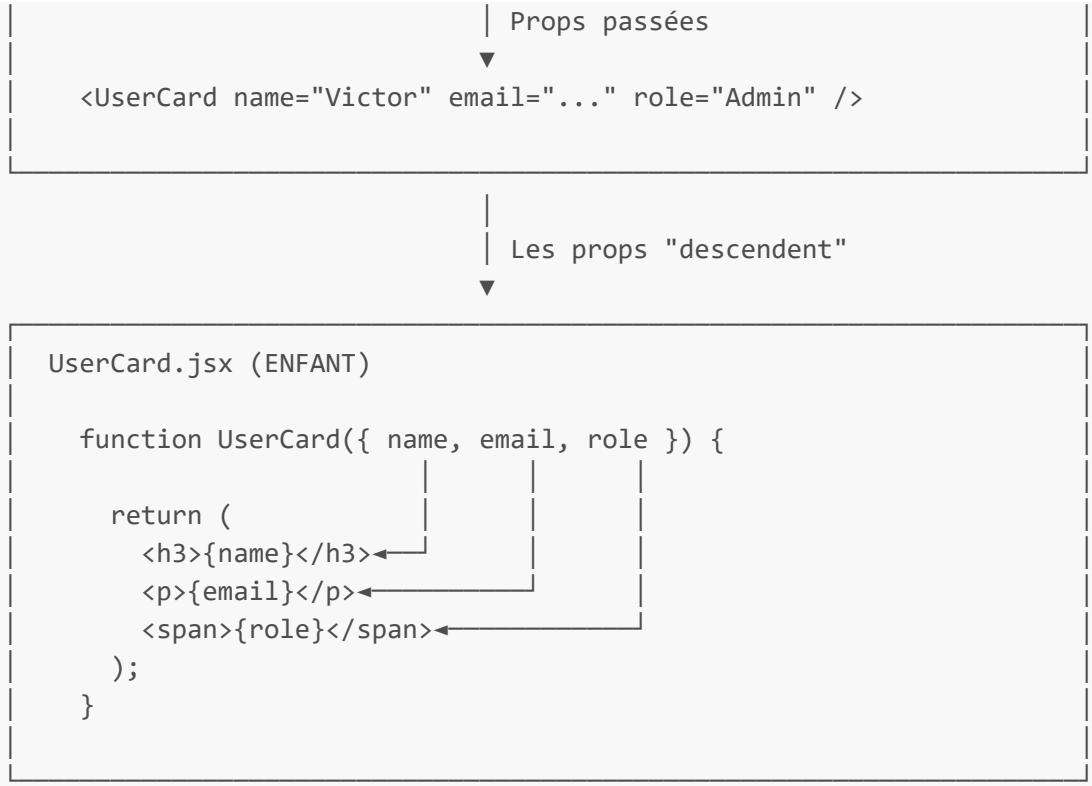
CSS associé

```
/* src/exercices/ex05-carte-utilisateur/UserCard.css */

.user-list-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
  gap: 1rem;
}

.user-card {
  border: 1px solid #e5e7eb;
  border-radius: 8px;
  overflow: hidden;
  background: white;
  box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1);
}

.user-card-header {
  display: flex;
  align-items: center;
  justify-content: space-between;
  padding: 1rem;
  background: #f9fafb;
}

.user-avatar {
  width: 60px;
  height: 60px;
  border-radius: 50%;
  object-fit: cover;
```

```
}

.user-status {
  font-size: 0.75rem;
  padding: 0.25rem 0.5rem;
  border-radius: 9999px;
}

.status-online {
  background: #d1fae5;
  color: #065f46;
}

.status-offline {
  background: #f3f4f6;
  color: #6b7280;
}

.user-card-body {
  padding: 1rem;
}

.user-name {
  margin: 0 0 0.25rem 0;
  font-size: 1.125rem;
}

.user-email {
  margin: 0 0 0.5rem 0;
  color: #6b7280;
  font-size: 0.875rem;
}

.user-role {
  display: inline-block;
  background: #e0e7ff;
  color: #3730a3;
  padding: 0.25rem 0.75rem;
  border-radius: 4px;
  font-size: 0.75rem;
  font-weight: 500;
}
```

Schéma mental : Flux des Props

```
┌──────────────────────────────────────────────────────────────────┐
│  UserList.jsx (PARENT)                                           │
│                                                                  │
│    const users = [{ name: "Victor", email: "..." }, ...]         │
│                                  │                               │
```



Pour aller plus loin

1. Ajouter une prop `onClick` pour rendre la carte cliquable
2. Créer un composant `Badge` réutilisable pour le rôle
3. Utiliser le spread operator pour passer toutes les props

```
// Prop callback
function UserCard({ name, onClick }) {
  return (
    <article onClick={() => onClick(name)}>
      ...
    </article>
  );
}

// Utilisation
<UserCard
  name="Victor"
  onClick={(name) => alert(`Clic sur ${name}`)}
/>

// Spread operator
{users.map(user => (
  <UserCard key={user.id} {...user} />
))}
// Équivalent à :
// <UserCard key={user.id} name={user.name} email={user.email} ... />
```