# Cheat Sheet - Rendu Conditionnel & Listes

## Rendu Conditionnel

### 1. Opérateur ternaire `? :`

**Quand l'utiliser** : Afficher A **ou** B selon une condition

```
// Syntaxe
{condition ? <SiVrai /> : <SiFaux />}

// Exemples
function UserStatus({ isLoggedIn }) {
  return (
    <div>
      {isLoggedIn ? <p>Bienvenue !</p> : <p>Connectez-vous</p>}
    </div>
  );
}

// Avec du texte
<span>{isActive ? "Actif" : "Inactif"}</span>

// Avec des classes CSS
<div className={isOpen ? "modal open" : "modal closed"}>
```

### 2. Opérateur `&&` (short-circuit)

**Quand l'utiliser** : Afficher **seulement si** condition vraie

```
// Syntaxe
{condition && <Element />}

// Exemples
function Notifications({ count }) {
  return (
    <div>
      {count > 0 && <span className="badge">{count}</span>}
    </div>
  );
}

// Afficher un message d'erreur
{error && <p className="error">{error}</p>}

// Afficher un bouton admin
{isAdmin && <button>Supprimer</button>}
```

### 3. Instructions `if` classiques

**Quand l'utiliser** : Logique complexe, plusieurs conditions

```
function UserGreeting({ user, isLoading, error }) {
  // Early return pour les cas spéciaux
  if (isLoading) {
    return <p>Chargement...</p>;
  }

  if (error) {
    return <p className="error">{error}</p>;
  }

  if (!user) {
    return <p>Aucun utilisateur</p>;
  }

  // Cas normal
  return <h1>Bienvenue {user.name} !</h1>;
}
```

### 4. Variable JSX pré-calculée

```
function StatusBadge({ status }) {
  let badge;

  if (status === 'success') {
    badge = <span className="green">Succès</span>;
  } else if (status === 'warning') {
    badge = <span className="orange">Attention</span>;
  } else if (status === 'error') {
    badge = <span className="red">Erreur</span>;
  } else {
    badge = <span className="gray">Inconnu</span>;
  }

  return <div className="status">{badge}</div>;
}
```

### Tableau comparatif

| Méthode | Cas d'usage | Exemple |
|---|---|---|
| Ternaire `? :` | A ou B | `{isOpen ? "Ouvert" : "Fermé"}` |
| `&&` | Afficher ou rien | `{hasError && <Error />}` |
| `if` + return | Plusieurs cas, early return | Chargement, erreur, vide |
| Variable JSX | Switch/case, logique complexe | Statuts multiples |

---

## Piège du `&&` avec les nombres

```
const count = 0;

// PIÈGE : affiche "0" dans le DOM !
{count && <span>Notifications: {count}</span>}

// Car : 0 && "texte" = 0 (falsy mais affiché)

// SOLUTIONS :
{count > 0 && <span>Notifications: {count}</span>}
{count !== 0 && <span>Notifications: {count}</span>}
{Boolean(count) && <span>Notifications: {count}</span>}
{!!count && <span>Notifications: {count}</span>}
```

---

## Listes avec `.map()`

### Syntaxe de base

```
const items = ['Pomme', 'Banane', 'Orange'];

function FruitList() {
  return (
    <ul>
      {items.map((fruit, index) => (
        <li key={index}>{fruit}</li>
      ))}
    </ul>
  );
}
```

### La prop `key` - OBLIGATOIRE

```
// Avec un ID unique (RECOMMANDÉ)
{users.map(user => (
  <UserCard key={user.id} user={user} />
))}

// Avec l'index (ACCEPTABLE si liste statique)
{colors.map((color, index) => (
  <span key={index}>{color}</span>
))}
```

### Pourquoi la `key` ?

React utilise `key` pour identifier chaque élément et optimiser les mises à jour.

```
// SANS key stable : React recrée tout à chaque changement
// AVEC key stable : React sait quel élément a changé

// Liste avant
<li key="1">Alice</li>
<li key="2">Bob</li>

// Liste après (ajout de Charlie au début)
<li key="3">Charlie</li>  // Nouveau
<li key="1">Alice</li>    // React sait que c'est le même
<li key="2">Bob</li>      // React sait que c'est le même
```

### Quand NE PAS utiliser l'index comme key

```
// PROBLÈME : liste modifiable (ajout, suppression, réordonnement)

const [tasks, setTasks] = useState([
  { id: 1, text: 'Tâche 1' },
  { id: 2, text: 'Tâche 2' },
]);

// MAUVAIS : l'index change quand on modifie la liste
{tasks.map((task, index) => (
  <Task key={index} task={task} />  // Bugs potentiels !
))}

// BON : ID stable
{tasks.map(task => (
  <Task key={task.id} task={task} />
))}
```

---

## Patterns courants avec les listes

### Filtrer avant d'afficher

```
function ActiveUsers({ users }) {
  return (
    <ul>
      {users
        .filter(user => user.isActive)
        .map(user => (
          <li key={user.id}>{user.name}</li>
        ))
      }
    </ul>
  );
}
```

### Trier avant d'afficher

```
function SortedList({ items }) {
  return (
    <ul>
      {[...items]  // Copie pour ne pas muter l'original
        .sort((a, b) => a.name.localeCompare(b.name))
        .map(item => (
          <li key={item.id}>{item.name}</li>
        ))
      }
    </ul>
  );
}
```

### Afficher un message si liste vide

```
function UserList({ users }) {
  if (users.length === 0) {
    return <p>Aucun utilisateur trouvé.</p>;
  }

  return (
    <ul>
      {users.map(user => (
        <li key={user.id}>{user.name}</li>
      ))}
    </ul>
  );
}

// Alternative avec ternaire
function UserList({ users }) {
  return (
    <>
      {users.length === 0 ? (
        <p>Aucun utilisateur trouvé.</p>
      ) : (
        <ul>
          {users.map(user => (
            <li key={user.id}>{user.name}</li>
          ))}
        </ul>
      )}
    </>
  );
}
```

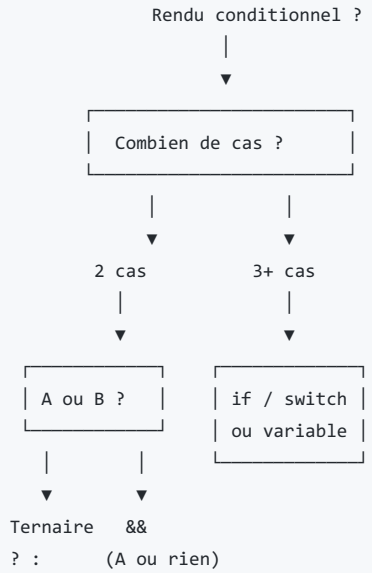
### Liste avec composant séparé

```
// UserCard.jsx
function UserCard({ user, onDelete }) {
  return (
    <div className="card">
      <h3>{user.name}</h3>
      <p>{user.email}</p>
      <button onClick={() => onDelete(user.id)}>Supprimer</button>
    </div>
  );
}

// UserList.jsx
function UserList({ users, onDeleteUser }) {
  return (
    <div className="user-list">
      {users.map(user => (
        <UserCard
          key={user.id}
          user={user}
          onDelete={onDeleteUser}
        />
      ))}
    </div>
  );
}
```

---

## Schéma mental : Choisir sa méthode




---

## Checklist Listes

- ☐ Chaque élément a une `key` unique
- ☐ La `key` est stable (pas l'index si liste modifiable)
- ☐ `key` est sur l'élément racine du map (pas un enfant)
- ☐ Gérer le cas "liste vide"
- ☐ Ne pas muter le tableau original (utiliser filter, map, spread)