

Cheat Sheet - Composants & Props

Qu'est-ce qu'un composant ?

Un composant React = **une fonction qui retourne du JSX**

```
// Composant minimal
function MonComposant() {
  return <h1>Hello</h1>;
}
```

Les 3 règles d'un composant

1. Nom en PascalCase (majuscule au début)

```
// OK
function UserCard() { ... }
function NavigationMenu() { ... }

// ERREUR - React ne reconnaîtra pas comme composant
function userCard() { ... } // minuscule = balise HTML
function navigation_menu() { ... }
```

2. Retourne du JSX (ou null)

```
function MonComposant() {
  return <div>Contenu</div>; // OK
}

function ComposantVide() {
  return null; // OK - n'affiche rien
}
```

3. Export pour être utilisable ailleurs

```
// Export par défaut (1 seul par fichier)
export default function MonComposant() { ... }

// Import
import MonComposant from './MonComposant';

// ---

// Export nommé (plusieurs possibles)
export function ComposantA() { ... }
export function ComposantB() { ... }

// Import
import { ComposantA, ComposantB } from './Composants';
```

Les Props : passer des données au composant

Analogie simple

Les props sont comme les **paramètres d'une fonction** ou les **attributs HTML**.

```
// HTML natif

  └─ attributs ─┘

// Composant React
<UserCard name="Victor" age={26} isAdmin={true} />
  └───────── props ─────────┘
```

Recevoir les props

```
// Méthode 1 : objet props
function UserCard(props) {
  return (
    <div>
      <h2>{props.name}</h2>
      <p>{props.age} ans</p>
    </div>
  );
}

// Méthode 2 : déstructuration (RECOMMANDÉ)
function UserCard({ name, age }) {
  return (
    <div>
      <h2>{name}</h2>
      <p>{age} ans</p>
    </div>
  );
}

// Utilisation
<UserCard name="Victor" age={26} />
```

Types de props

```
// String : guillemets
<Composant titre="Bonjour" />

// Number : accolades
<Composant age={26} />

// Boolean : accolades (ou juste le nom pour true)
<Composant isActive={true} />
<Composant isActive /> // équivalent à isActive={true}
<Composant isActive={false} />

// Objet : accolades + accolades
<Composant user={{ name: "Victor", age: 26 }} />

// Tableau : accolades
<Composant items={[1, 2, 3]} />

// Fonction : accolades
<Composant onClick={handleClick} />
<Composant onClick={() => console.log('click')} />
```

Props par défaut

```
// Méthode 1 : dans la déstructuration (RECOMMANDÉ)
function Button({ label = "Cliquez", color = "blue" }) {
  return <button style={{ color }}>{label}</button>;
}

// Utilisation
<Button /> // label="Cliquez", color="blue"
<Button label="Envoyer" /> // label="Envoyer", color="blue"
```

Props spéciale : children

`children` = tout ce qui est entre les balises ouvrante et fermante

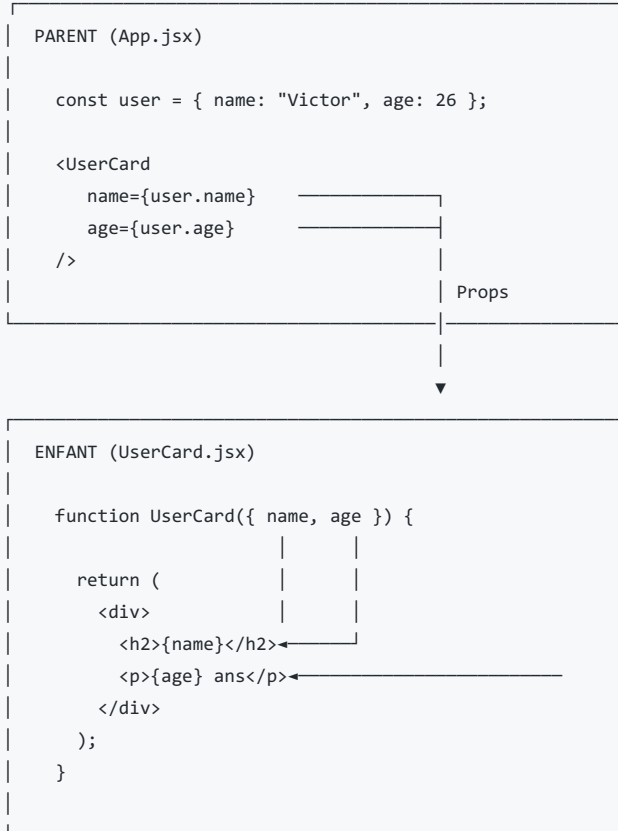
```
function Card({ children, title }) {
  return (
    <div className="card">
      <h2>{title}</h2>
      <div className="card-body">
        {children}
      </div>
    </div>
  );
}

// Utilisation
<Card title="Ma carte">
  <p>Ceci est le contenu</p>
  <button>Action</button>
</Card>
```

Résultat :

```
<div class="card">
  <h2>Ma carte</h2>
  <div class="card-body">
    <p>Ceci est le contenu</p>
    <button>Action</button>
  </div>
</div>
```

Schéma mental : Flux des données



Règle d'or : Props en lecture seule !

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

  // OK : utiliser la prop pour créer une nouvelle valeur
  const displayName = name.toUpperCase();

  return <h2>{displayName}</h2>;
}
```

Les props **descendent** du parent vers l'enfant. Pour remonter des données : utiliser une **fonction callback** en prop.

Pattern : Fonction callback en prop

```
// Parent
function App() {
  function handleClick(userId) {
    console.log("User cliqué:", userId);
  }

  return <UserCard id={42} onClick={handleClick} />;
}

// Enfant
function UserCard({ id, onClick }) {
  return (
    <button onClick={() => onClick(id)}>
      Voir profil
    </button>
  );
}
```

Checklist composant

- ☐ Nom en PascalCase
- ☐ Fichier .jsx avec le même nom
- ☐ Export (default ou nommé)
- ☐ Props déstructurées
- ☐ Props en lecture seule
- ☐ Un seul élément parent retourné