***REACT NOTES***

1. **CONCEPTS DE BASE DE REACT JS :**

**Composants**

React est basé sur l'idée de composants réutilisables. Un composant React est une unité indépendante de l'interface utilisateur, qui est souvent encapsulée et qui peut être réutilisée partout dans l'application.

**JSX**

JSX est une syntaxe qui ressemble à du HTML et qui est utilisée pour décrire l'aspect visuel des composants React. Elle est transpilée en appels JavaScript purs, généralement grâce à Babel.

**État (State) et Props**

Les composants React peuvent gérer un "état" local, qui est un objet JavaScript contenant des données ou des informations. Les "props" sont des paramètres que les composants parent passent aux composants enfant.

**Cycle de vie des Composants**

Les composants React ont un cycle de vie qui permet d’exécuter du code à certains moments clés de la vie du composant, comme lors de sa création, sa mise à jour, ou sa destruction.

**Hooks**

Introduits dans React 16.8, les hooks permettent d'utiliser l'état et d'autres fonctionnalités de React sans écrire de classes. Les hooks les plus couramment utilisés sont **useState** et **useEffect**.

**Événements Synthétiques**

React encapsule les événements natifs du navigateur dans des événements synthétiques pour assurer la cohérence des événements dans les différents navigateurs.

**Virtual DOM**

React crée une copie légère du DOM appelée Virtual DOM. React utilise le Virtual DOM pour optimiser les mises à jour du DOM, en ne modifiant le DOM réel que lorsque c'est nécessaire.

**Clés (Keys)**

React utilise les clés pour identifier les éléments uniques dans les listes et gérer efficacement les modifications, les ajouts ou les suppressions d'éléments.

**Composants Contrôlés**

Dans un formulaire HTML, les éléments tels que **<input>**, **<textarea>**, et **<select>** maintiennent habituellement leur propre état. React permet de transformer ces éléments en composants contrôlés, où React contrôle la valeur de ces éléments de formulaire.

**Levage d'État (Lifting State Up)**

Parfois, plusieurs composants ont besoin de refléter le même état changeant. React propose une solution pour cela en levant l'état vers le parent le plus proche commun.

**Flux de Données Unidirectionnel**

React est conçu pour un flux de données unidirectionnel, ce qui rend le comportement de l'application plus prévisible et plus facile à comprendre.

**Performance et Optimisation**

Comprendre comment utiliser **React.memo**, **useCallback**, **useMemo**, et comment effectuer le fractionnement de code (code-splitting) avec **React.lazy** et **Suspense** peut être crucial pour améliorer les performances des applications React.