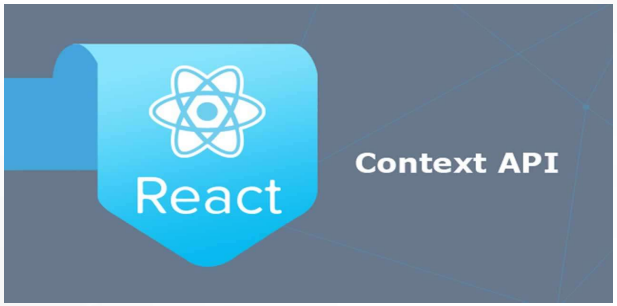
Chapitre 9 : L'API Context

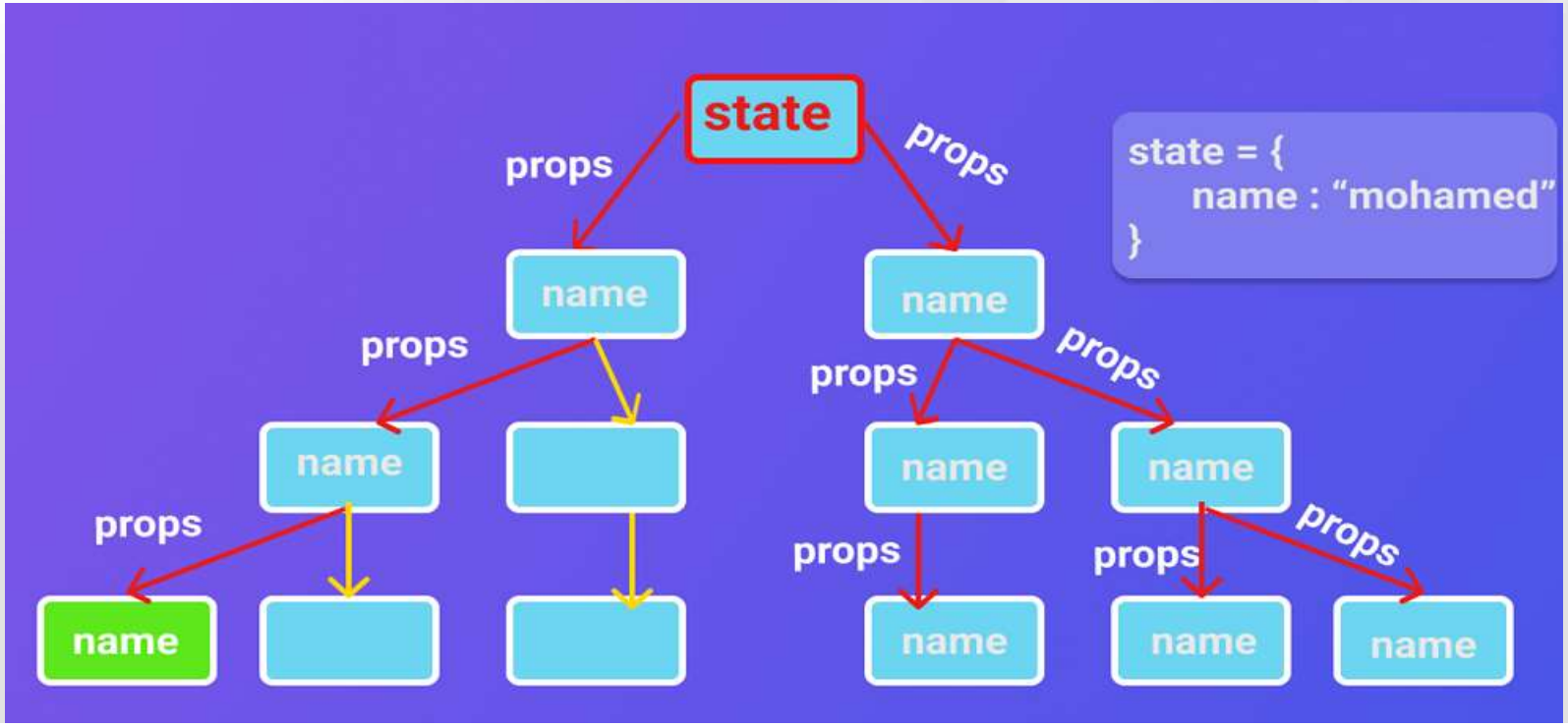


Introduction

- L'API de contexte est une structure de composant fournie par React pour gérer les états de tous les niveaux de l'application.
- Il permet d'éviter d'utiliser des bibliothèques tierces pour l'application.
- Pour sa mise en place, il faut définir le contexte et on doit juste passer les valeurs de contexte via le fournisseur pour l'application de react.
- Si chaque composant gère son propre état, on a les props pour transmettre les données mais cela ne fonctionne que dans le cas de la relation parent-enfant.
- Mais lorsque la base de code se compose de nombreux composants qui dépendent d'un seul élément de données, mais qui sont imbriqués profondément dans l'arborescence des composants, l'utilisation de Contexte est la meilleure solution dans ce cas d'utilisation,
- Le Context envoie des propriétés sans passer par un intermédiaire.

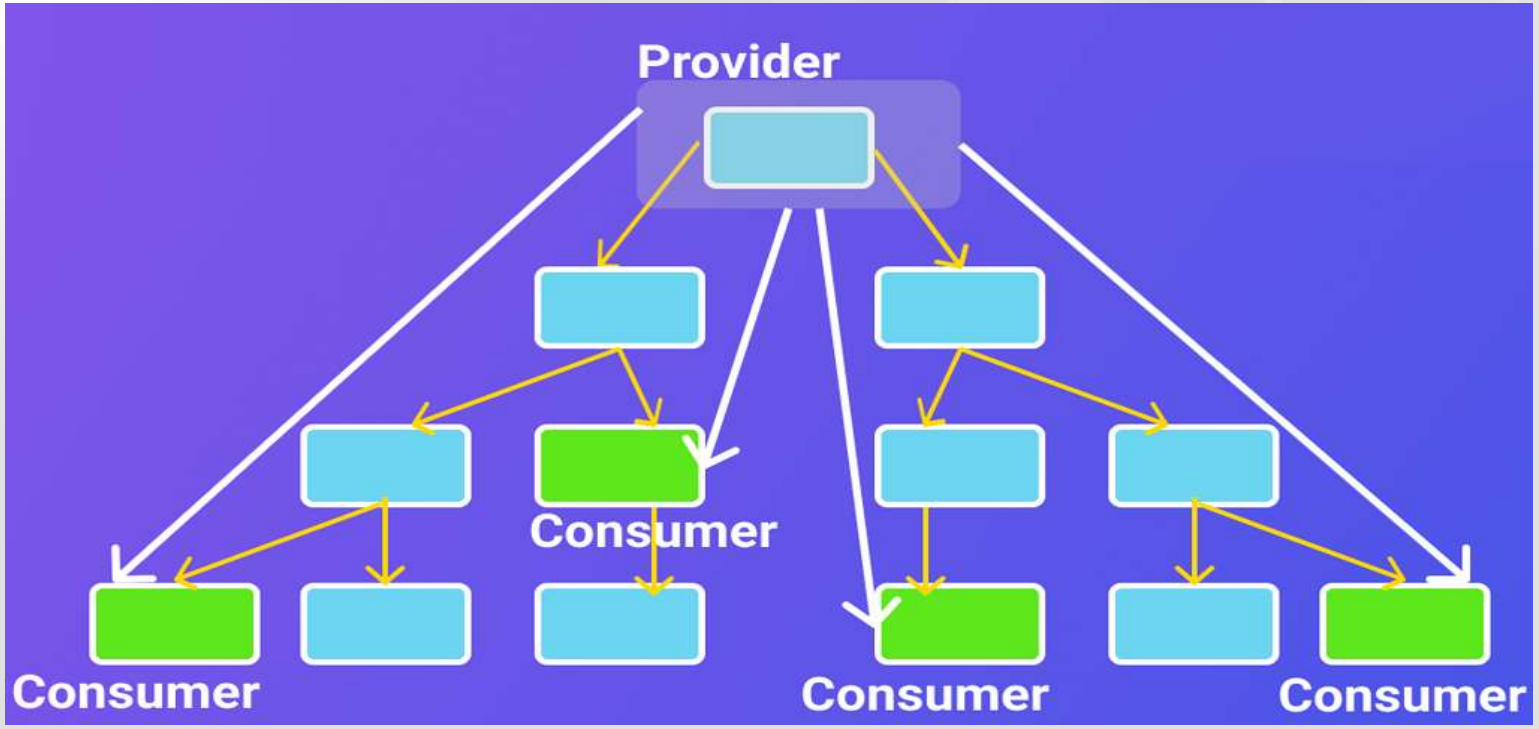
Sans Context

- Nous avons élevé l'état au composant le plus élevé et avons passé les données à travers plusieurs composants qui n'avaient pas besoin à l'origine de ces informations et c'est uniquement pour les transmettre à un composant qui en a besoin.



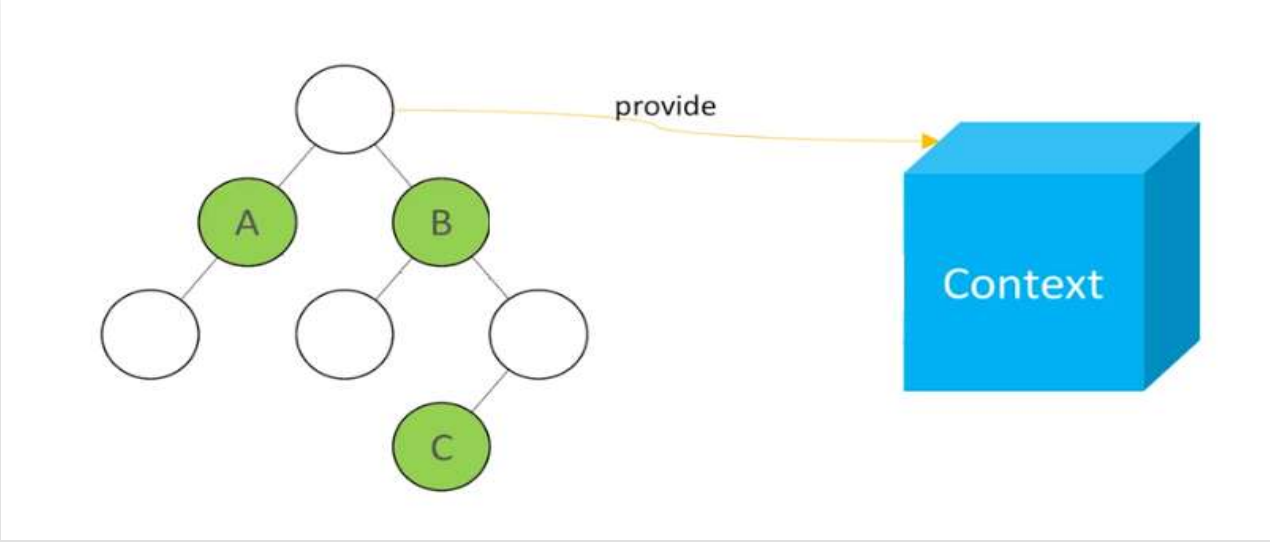
Avec Context

- L'API context a un seul but précis : envoyer des propriétés vers les composants enfants depuis n'importe quels composants parents, sans forcément passer par un intermédiaire.



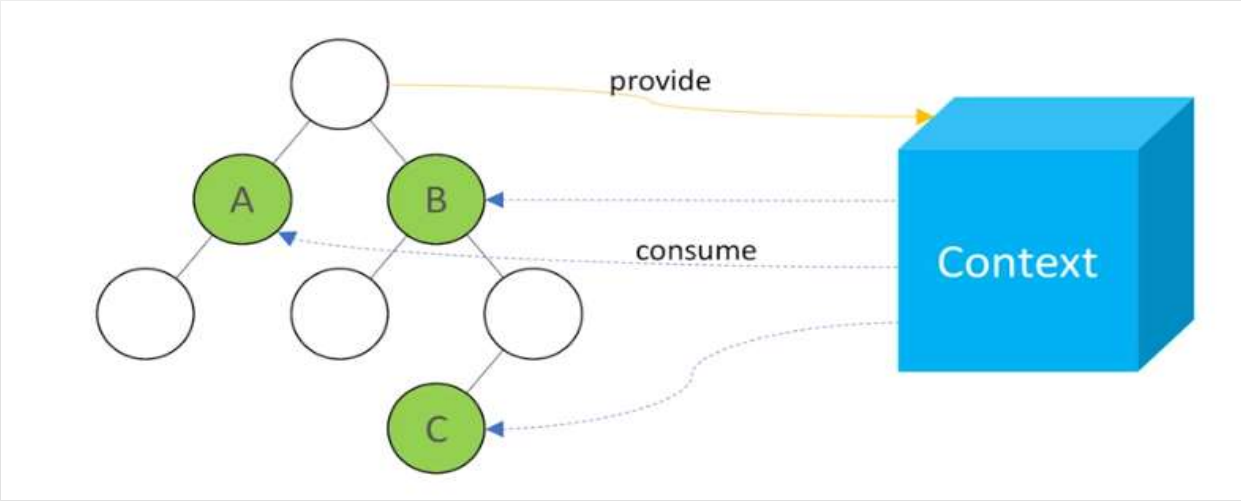
Provider

- Le composant Provider est utilisé dans la hiérarchie supérieure de l'arborescence.
- Il accepte un accessoire appelé Value.
- Il agit comme un composant racine dans l'arborescence hiérarchique de sorte que tout enfant de l'arborescence puisse accéder aux valeurs fournies par le fournisseur de contexte.



Consumer

- Comme son nom l'indique, le Consumer consomme les données qui sont transmises, quelle que soit la profondeur de leur imbrication dans l'arborescence des composants.
- Cela signifie que le Consumer ne doit pas nécessairement être l'enfant du fournisseur.
- Au lieu de cela, il peut accéder aux données de n'importe où dans l'arborescence des composants.
- Un Consumer rend les données à l'aide d'une API de prop de rendu.



Création d'un Context 1/2

- L'objet Context est créé avec la méthode createContext de React.
- L'objet résultant sera utilisé dans tous les composants.
- Ensuite, on définit une fonction qui nous aidera à distribuer les données via le Provider.

```
import React, { createContext, useState } from "react"
export const ArticleContext = createContext();

const ArticleProvider = ({ children }) => {
  const [articles, setArticles] = useState([
    { id: 1, title: "post 1", body: "content 1" },
    { id: 2, title: "post 2", body: "content 2" }
     ])
```

Création d'un Context 2/2

```
const saveArticle = article => {
  const newArticle = {
    id: Math.random(),
    title: article.title,
    body: article.body,
  }
  setArticles([...articles, newArticle])
 }
 return (
  <ArticleContext.Provider value={{ articles, saveArticle }}>
    {children}
  </ArticleContext.Provider>
 )
}
```

Fournir un Context

- Une fois le contexte créé, on pourrait désormais le fournir à tous les composants qui devraient pouvoir interagir avec lui.
- Le contexte doit être fourni dans un composant qui encapsule tous les composants enfants qui ont éventuellement besoin d'accéder au contexte. A travers <MyContext.Provider>.
- Pour les données qui devraient être disponibles dans l'ensemble de l'application, on doit donc fournir le contexte dans le composant racine (par exemple, <App />).

```
import React from "react"
import ArticleProvider from "./context/articleContext"
import Articles from "./containers/Articles"
import AddArticle from "./components/AddArticle/AddArticle"
function App() {
 return (
  <ArticleProvider>
   <AddArticle />
   <Articles />
  </ArticleProvider>
 )  }  export default App;
```

Consommer un Context 1/3

- Avec React hooks, on utilise l'accès au hook useContext qui nous aidera à consommer le contexte.
- useContext accepte un objet contexte (la valeur renvoyée par React.createContext), et renvoie la valeur actuelle du contexte.
- Celle-ci est déterminée par la prop value du plus proche <MyContext.Provider> au-dessus du composant dans l'arbre.

```
import React, { useState, useContext } from "react"
import "./AddArticle.css"
import { ArticleContext } from "../../context/articleContext"

const AddArticle = () => {
  const { saveArticle } = useContext(ArticleContext)
  const [article, setArticle] = useState()
  const handleArticleData = e => {
    setArticle({
      ...article,
      [e.target.id]: e.target.value,
    })   }
```

Consommer un Context 2/3

```
const addNewArticle = e => {
  e.preventDefault()
  saveArticle(article)
}
return (
  <form onSubmit={addNewArticle} className="add-article">
    <input
      type="text"
      id="title"
      placeholder="Title"
      onChange={handleArticleData}
    />
```

Consommer un Context 3/3

```
 <input
       type="text"
       id="body"
       placeholder="Body"
       onChange={handleArticleData}
      />
      <button>Add article</button>
    </form>
  )
}
export default AddArticle
```