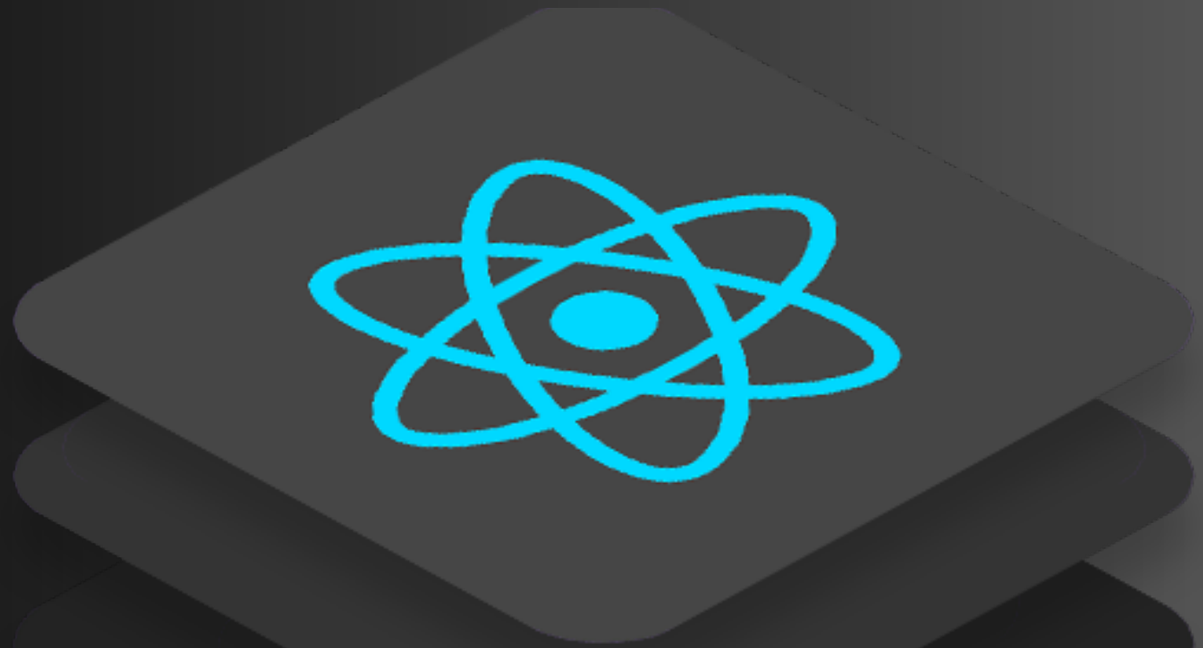


React

Frontend

Context API

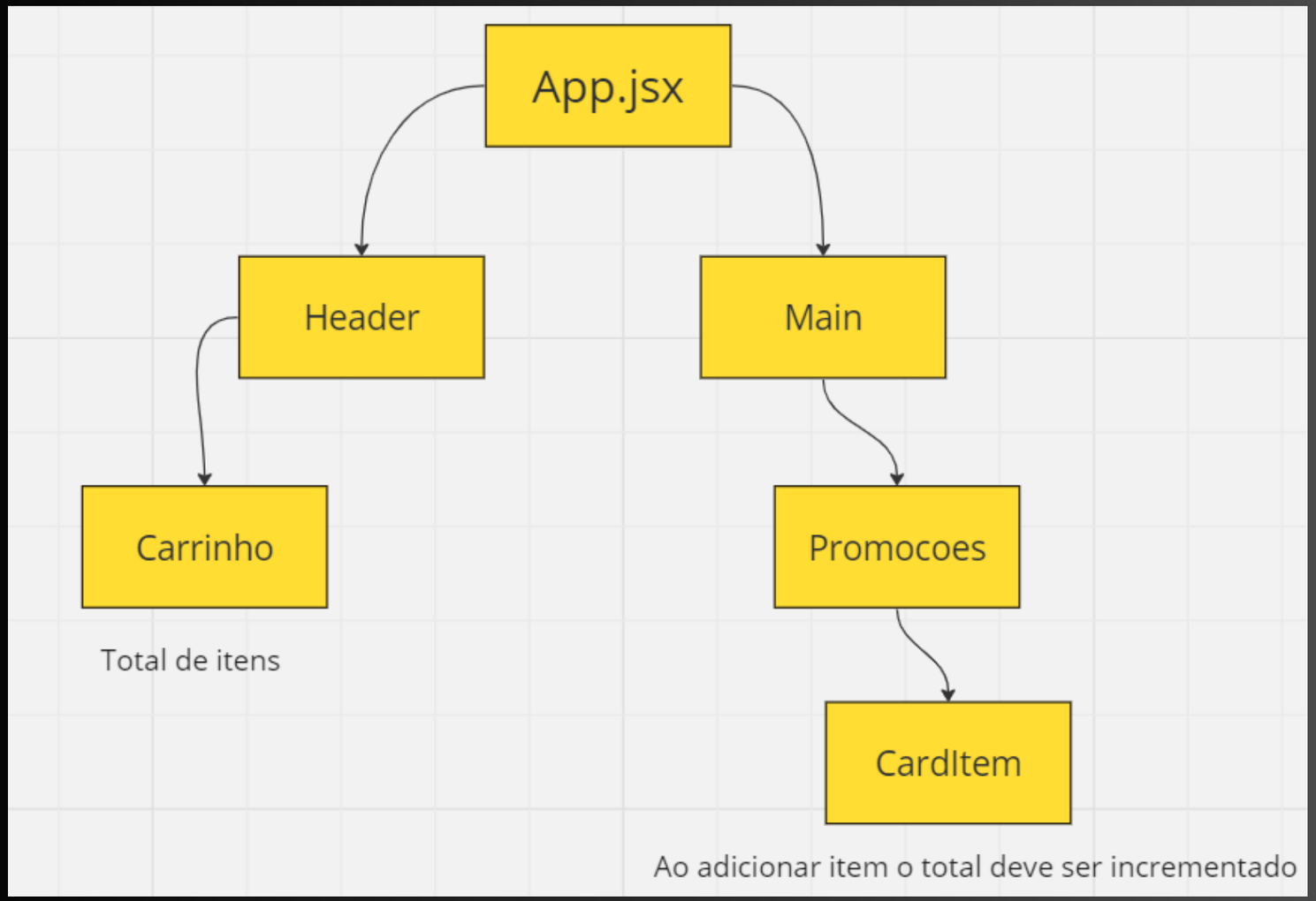
Contexto (context) disponibiliza uma forma de passar dados entre a árvore de componentes sem precisar passar props manualmente em cada nível.

Em uma aplicação típica do React, os dados são passados de cima para baixo (de pai para filho) via props, mas esse uso pode ser complicado para certos tipos de props (como preferências locais ou tema de UI), que são utilizadas por muitos componentes dentro da aplicação.

Contexto (context) fornece a forma de compartilhar dados como esses, entre todos componentes da mesma árvore de componentes, sem precisar passar explicitamente props entre cada nível.

Contextualização e Justificativa

Carrinho de compras



Precisamos de um estado para gerenciar o carrinho

Problema: CardItem e Carrinho tem como “parentesco” mais próximo o App.jsx

Para que o estado esteja acessível ao Carrinho e ao CardItem , utiliza-se de lift

Lifting

No React, o lift é o processo de mover dados de um componente filho para um componente pai. Isso é feito para compartilhar dados entre componentes que não estão diretamente conectados.

O lift é uma técnica comumente usada para compartilhar dados globais, como o estado da aplicação ou a configuração do usuário. Também pode ser usado para compartilhar dados específicos entre componentes que não estão diretamente conectados.

Lifting

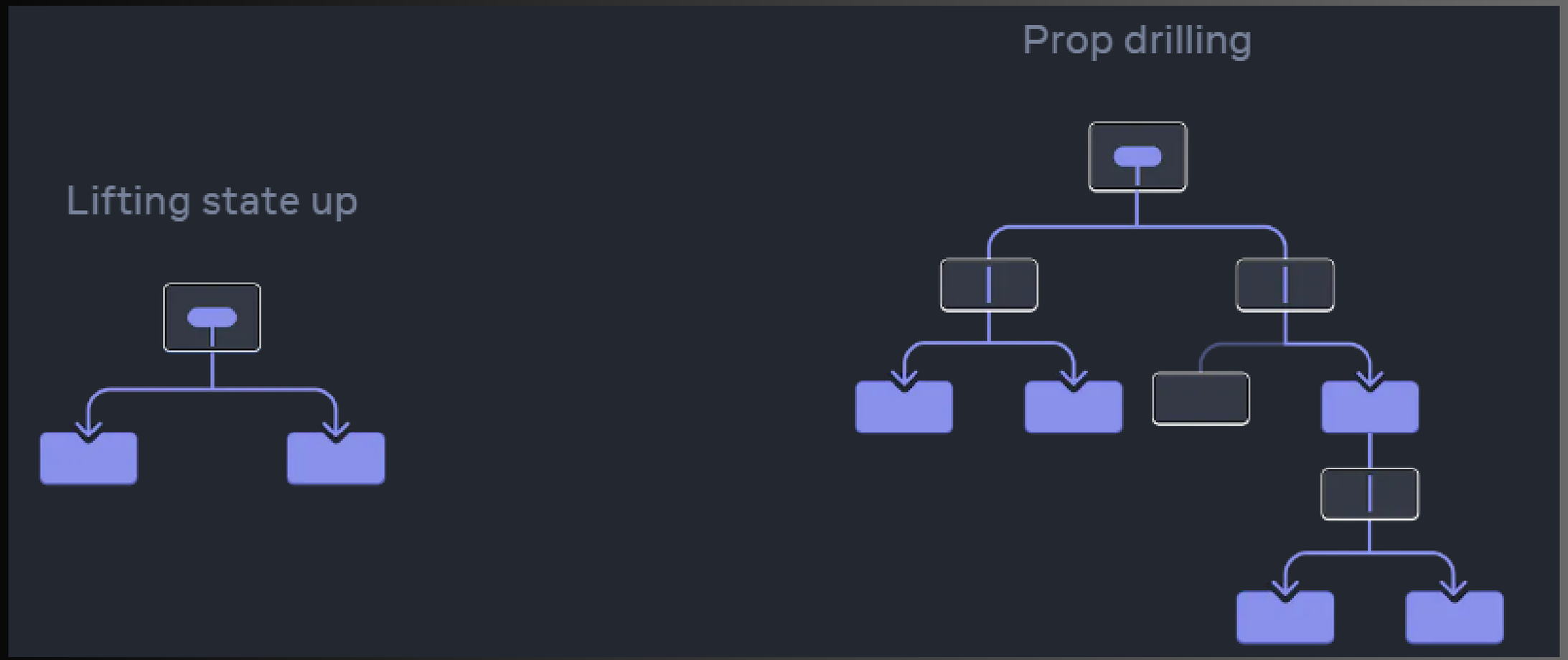
Existem duas maneiras principais de realizar o lift no React:

Usando props: As props são uma maneira de passar dados de um componente pai para um componente filho. Para usar props para realizar o lift, basta definir o dado que deseja compartilhar como uma prop no componente pai. O componente filho pode então acessar o dado usando o método this.props.
(Pode gerar o problema de Prop Drilling)

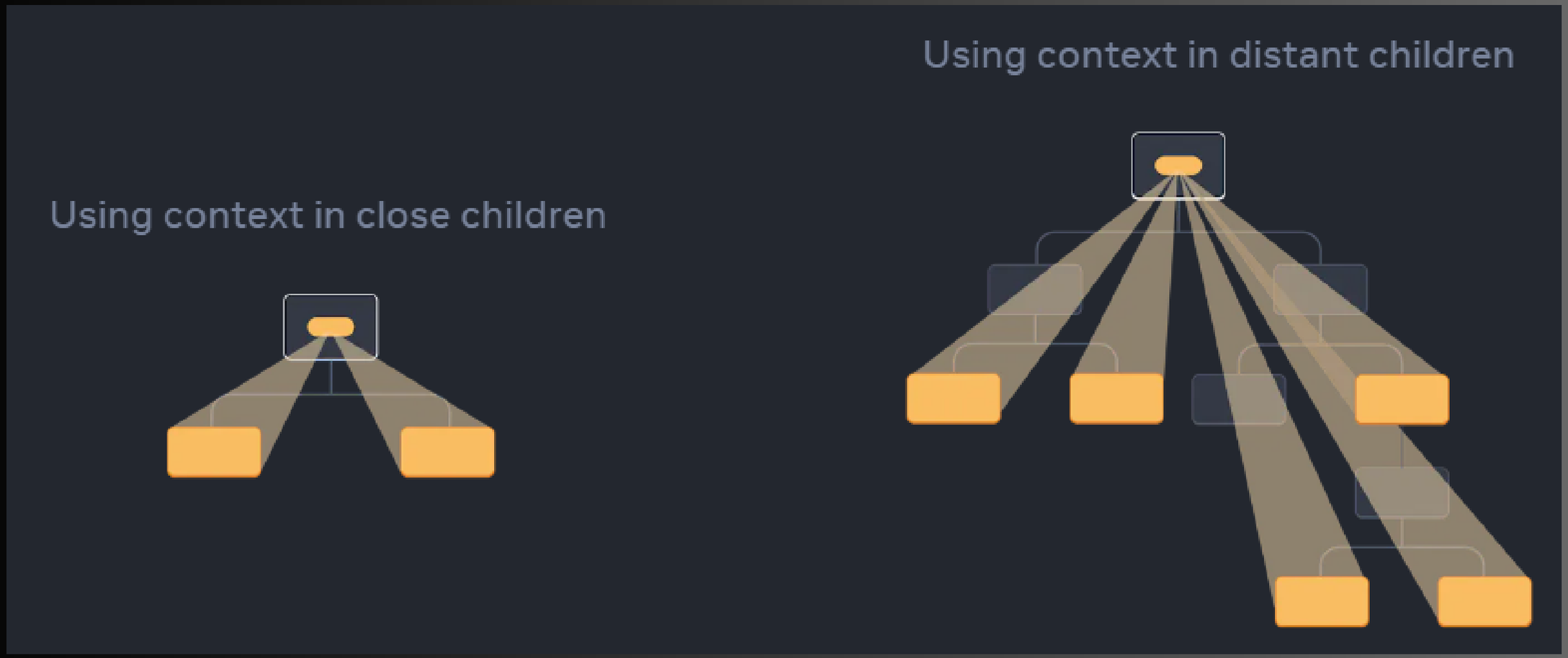
Usando o Context API: O Context API é uma maneira de compartilhar dados globalmente. Para usar o Context API para realizar o lift, basta criar um Contexto e um provider. O provider fornece o dado que deseja compartilhar, e o Contexto é usado para consumir o dado.

Prop Drilling

Em React, o "prop drilling" se refere a uma técnica (às vezes considerada antipadrão) de passar dados de um componente pai para seus filhos através das props, camada por camada, até chegar ao componente que realmente precisa da informação. É como se a propriedade fosse "perfurada" de um componente até o outro, descendo pela árvore de componentes.



Solução

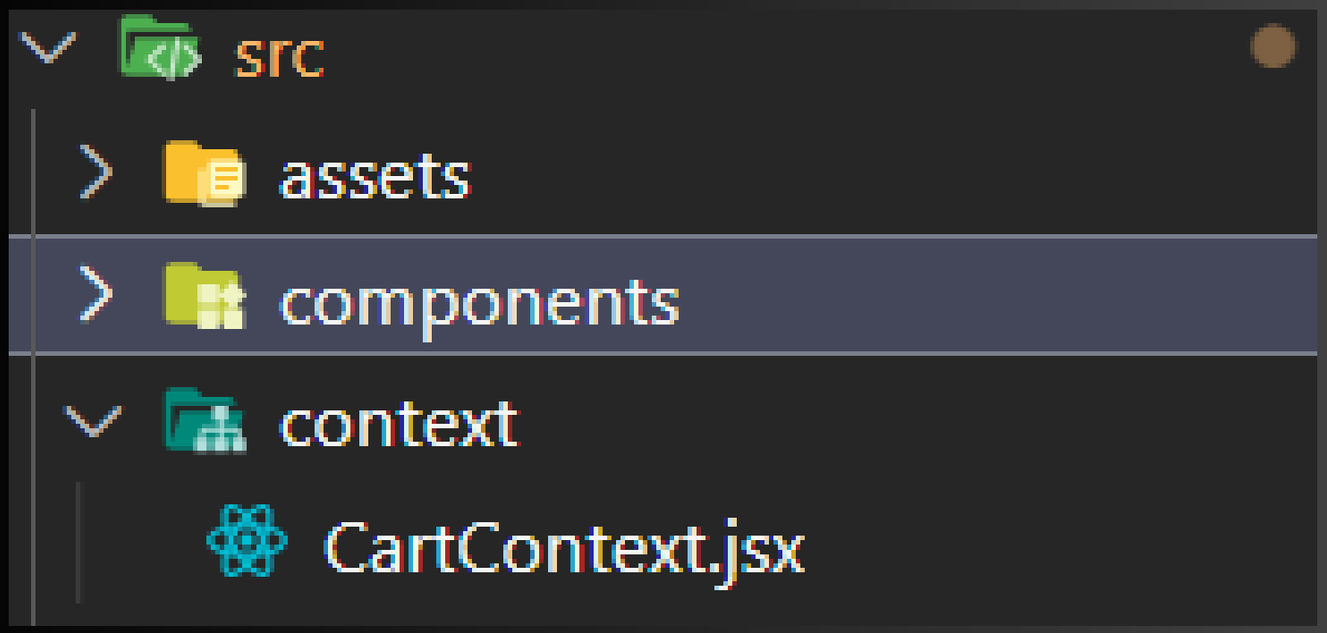


Com o Context API, você pode compartilhar dados globalmente, sem a necessidade de passar props de componente em componente. Assim, você transporta a informação diretamente ao componente desejado.

Criando Context

Normalmente, cria-se uma pasta context para alocar os contextos criados

Se o contexto é criado para disponibilzar o estado global de um Cart, então, nomeia-se esse contexto de CartContext



Para criar o contexto, você deverá utilizar createContext(). Para disponibilizá-lo, você deverá exportar a constante que armazena o contexto criado.

```
import { createContext} from "react";

export const CartContext = createContext([])
```

O valor passado dentro de createContext é o valor inicial na montagem dos componentes.

Provider

```
export function CartProvider({children}){

    return(
        <CartContext.Provider>
            {children}
        </CartContext.Provider>
    )
}
```

Para fornecer o contexto de forma global, cria-se um componente provider que vai receber os componentes como filhos e assim, criar um contexto(escopo) comum a todos.

Fornecendo o contexto de forma global

```
ReactDOM.createRoot(document.getElementById('root')).render(
    <CartProvider>
      <App />
    </CartProvider>
)
```

Esse componente provider deve envolver todos os componentes que necessitam da informação fornecida pelo contexto. A maneira mais prática de fazer isso é envolver o componente pai de todos, o componente App.

useContext() - Utilizando o contexto

Para utilizar o contexto, basta inseri-lo no componente desejado e utilizá-lo através do useContext.

```
import { useContext, useEffect, useState } from "react"
import { CartContext } from "../../context/CartContext"
```

```
const meuContexto = useContext(CartContext)
console.log(meuContexto)
```

Dessa forma, o contexto tem comportamento estático. Se desejar que ele reaja como um estado, algumas mudanças deverão ser adotadas.

Adicionando estado ao contexto

```
export function CartProvider({children}){
    const[cart, setCart] = useState([])
    return(
        <CartContext.Provider value={{cart, setCart}}>
            {children}
        </CartContext.Provider>
    )
}
```

Adiciona-se o estado ao componente e então, é passado como props em value um objeto contento o estado e o método para modificar esse estado ({cart, setCart})

```
const {cart, setCart} = useContext(CartContext)
```

Para utilizar o estado e o método no componente, basta desestruturar o objeto a partir do useContext e o contexto criado do Cart.

Vamos à prática