Universidade Técnica do Atlântico
Departamento de Informática
Linguagem e Tecnologias Web

Relatório Técnico: Uso do Hook useReducer no Gerenciamento de Estados em React

Rodrigo Fortes

6-Dezembro-2024

Conteúdo

1 Introdução

A manipulação de estados em React é exencial no desenvolvimentos de aplicações no React, que garante o fluxo de dados e a movimentação desses dados dentro e fora dos componentes. Em React, ou em qualquer interface de desenvolvimento web , o desenvolvimento de aplicações se torma mais desafiador , gerenciarestados que envolvem múltiplas interações pode ser um grande problema. Para lidar com esses cenários, o react oferece o hook usereducer, que proporciona uma abordagem baseada em reducers para controlar estado de forma estruturada. Neste relatorio iremos citar os conceitos do hook usereducer, como é que funcionam, as suas caracteristicas e relaciona-lo com useState utilizando exemplos práticos para demonstrar seu uso em aplicações React

2 Funcionamento do Hook useReducer

2.1 Descrição Geral

O useReducer é um hook do React que permite adicionar um reducer a um componente para gerenciar estados complexos e ações de maneira estruturada e previsível. Ela é amplamente usado em cenários onde a lógica de atualização de estado é mais detalhada ou onde múltiplas interações precisam ser gerenciadas em um único componente funcionaal.

2.1.1 Por que usar o useReducer

O useReducer é especialmente útil em situações como:

1. **Estados com múltiplos valores interconectados**: Como os objetos com vários propriedades ou listas que precisam ser alteradas frequentemente.

2. **Centralização da lógica de atualização do estado**: mantendo a lõgica de alteração de estado separada das interaçõe do usuário.

3. **Componentes que precisam de previsibilidade nas alterações**: Útil para rastrear exatamente como e por que o estado foi alterado.

Comparado ao useState, o useReducer oferece maior controle e clareza para gerenciar estados complexos

2.1.2 Usereducer vs UseState

Embora o useReducer traga vantagenes em certos casos, ele tambémtem desvantagens. Abaixo entá uma comparação entre o usereducer e o useState:

1. **Tamanho do Código**: **useState** exige menos código inicial, enquanto o useReducer requer a criação de uma funcção reducer e ações. Contudo, o **useReducer** pode reduzir reduncdancias quando várias atualizações de estado seguem uma lógica semelhante.

2. **Legibilidade**: Em estados simples, o useState é mais facil de ler. Já o usereducer organiza melhor o código ao separar a logica de atualizações de definição dos eventos que causaram a mudança.

3. **Depuração**: O useReducer facilita o rastreamento de atualizações, permitindo adicionar logs noreducer para identificar ações e mudanças no estado, algo mais dificil com useState.

4. **Testabilidade**: O reducer, por ser uma função pura, pode ser testado de forma isolada, o que é útil para validar a lógica de estados complexos. O useState por outro lado, é testado diretamente no componente

5. **Recomendação e Preferência**: Use useReducer em estados complexos ou com bugs frequentes. Para estados simples, o **useState** é suficiente. Ambos podem ser combinados no mesmo componente, dependendo do caso.

2.2 Fluxo de Funcionamento

Nesta secção iremos apresentar como o useReducer funciona na maioria dos casos. Definindo o useReducer como um hook de gerenciamento de estados mais complexos, é um estado que é atualizado por meio de uma função reducer, que recebe o estado atual e uma ação e retorna o novo estado.

1. **Definição do Estado Inicial**: O estado inicial é configurado antes de usar o useReducer. Pode ser um valor simples, um objeto ou um array.

```
const initialState = { count: 0 };
```

Figura 1: Declaração do estado inicial

2. **A Função Reducer**: O reducer é uma função pura que recebe o estado atual e uma ação como argumentos, e retorna o novo estado com base no tipo da ação (action.type).

```
function reducer(state, action) {
    switch (action.type) {
        case 'INCREMENT':
            return { count: state.count + 1 };
        case 'DECREMENT':
            return { count: state.count - 1 };
        default:
            return state;
    }
}
```

Figura 2: Declaração da função reducer

Na função acima, o reducer é definido com dois parametros: o parametro **state** que representa o estado atual e o parametro **action** que é

um objeto que descreve a operação desejada. Geralmente contém uma propriedade type e, opcionalmente, outras informações (payload).

3. **Uso do useReducer**:O useReducer é chamado passando dois argumentos: o reducer e o estado inicial. Ele retorna dois valores: o estado atual (state) e a função dispatch para enviar ações ao reducer. Abaixo se encontra o sintaxe básica do useReducer:

```
const [state, dispatch] = useReducer(reducer, initialState);
```

Figura 3: Sintaxe básica do useReducer

Neste caso, o useReducer é declarado com o state, que contem o estado atual, a variavel dispatch que é usado para enviar ações ao reducer e iniciar atualizações de estado.

Exemplo de uso no componente:

```
function Counter() {
    const [state, dispatch] = useReducer(reducer, initialState);

    return (
        <div>
            <p>Contador: {state.count}</p>
            <button onClick={() => dispatch({ type: 'INCREMENT' })}>+</button>
            <button onClick={() => dispatch({ type: 'DECREMENT' })}>-</button>
        </div>
    );
}
```

Figura 4: Exemplo

Neste exemplo o estado é atualizado com base nas ações enviadas via dispatch e os botões enviam ações como type: 'INCREMENT' ou type: 'RESET' .

3 Conclusão

O useReducer é uma ferramenta super útil para gerenciar estados complicados em um aplicativo React. Ele se encaixa perfeitamente quando os estados são interdependentes na confluência de múltiplas interações. Além disso, a lógica de atualização é ainda mais concentrada em um lugar — uma função pura — o que significa que é muito fácil de suportar, depurar e testar. Geralmente requer mais código inicial do que useState, mas sempre aumentará a previsibilidade e a manutenibilidade. Para estados simples, useState permanecerá mais confortável, o que enfatiza a flexibilidade do React em se ajustar a cada necessidade. Dito isso, useReducer é uma opção mais poderosa e mais estruturada para aplicativos modernos e escaláveis.