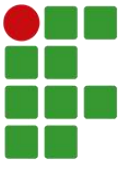

INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Passo Fundo

Artur Mariano da Silva

Trabalho de Pesquisa: React Hooks

Passo Fundo, abril de 2024

Artur Mariano da Silva

Trabalho de Pesquisa: React Hooks

Trabalho da disciplina de Programação Web.

Professor: Jorge Luis Boeira Bavaresco

Passo Fundo, abril de 2024

Introdução

O React é hoje uma das bibliotecas mais utilizadas para o desenvolvimento de interfaces de usuário, conforme consta na própria documentação. Ele provê inúmeras ferramentas aos desenvolvedores, sendo extremamente completo e atendendo às mais diversas necessidades.

O React trabalha com conceitos de componentes e estados, os quais são manipulados e fazem com que essa biblioteca seja hoje tão difundida. Para isso, é necessário que se conheça os *Hooks*. Eles se tornam extremamente importantes para a manipulação de estados, ao passo que permitem o uso desses estados sem que seja necessário escrever uma nova classe. Alguns exemplos são: useEffect, *useState*, *useContext*, *useReducer*, *useRef*, *useMemo* e *useCallback*.

Nesse sentido, no decorrer das aulas da disciplina de Programação Web tivemos contato com os *hooks useEffect*, *useState* e *useContext*. Por isso, o presente trabalho terá como objetivo focar nos *hooks* restantes diante dos citados acima - *useReducer*, *useRef*, *useMemo* e *useCallback* - visando um aprofundamento e um complemento dos estudos desenvolvidos no decorrer das aulas.

useReducer

O hook useReducer é uma função do React que permite gerenciar o estado de um componente de forma mais flexível e escalável, especialmente em casos em que o estado possui uma estrutura mais complexa e requer lógica de atualização sofisticada.

Ao utilizar *useReducer*, define-se um estado inicial e uma função reducer que especifica como o estado deve ser atualizado em resposta a diferentes tipos de ações. Essas ações são geralmente objetos que contêm um tipo (ou identificador) e, opcionalmente, dados adicionais.

A função *reducer* recebe o estado atual e a ação como argumentos e retorna um novo estado, sem modificar o estado original. Essa abordagem imutável garante previsibilidade e facilita o rastreamento de mudanças no estado.

Além disso, o *useReducer* oferece a capacidade de agrupar lógica de atualização relacionada em um único lugar, tornando o código mais organizado e fácil de manter.

Por fim, denota-se que o hook *useReducer* é uma ferramenta poderosa para o gerenciamento de estados no React, sendo especialmente útil em cenários complexos onde o uso de *useState* pode se tornar inadequado. Ele oferece uma abordagem mais estruturada e declarativa para lidar com a evolução do estado da aplicação.

useCallback

O hook *useCallback* atua diretamente na otimização de desempenho do React, especialmente quando se trata de evitar renderizações desnecessárias de componentes. Ele permite memorizar funções *callback*, garantindo que essas funções sejam recriadas apenas quando alguma de suas dependências mudar.

O hook recebe uma função *callback* e uma matriz de dependências como parâmetros, e retorna uma versão memorizada da função. Isso significa que, se as dependências não mudarem entre renderizações, o *useCallback* retornará a mesma instância da função, evitando a necessidade de recriá-la.

Essa memorização de funções é fundamental para evitar a criação repetida de instâncias de funções em cada renderização, o que pode consumir recursos

desnecessários e impactar negativamente o desempenho do aplicativo.

O uso do *useCallback* é especialmente útil quando se trabalha com componentes filhos que recebem funções como props. Ao memorizar essas funções com *useCallback*, garante-se que os componentes filhos não sejam re-renderizados desnecessariamente sempre que o componente pai for renderizado, a menos que as dependências da função tenham mudado.

Com isso, o *useCallback* é uma ferramenta valiosa para otimizar o desempenho do React, garantindo que as funções *callback* sejam recriadas apenas quando necessário, e não em todas as renderizações, melhorando assim a eficiência e a responsividade do aplicativo.

useMemo

O hook *useMemo* é um hook do React que atua na otimização de desempenho do React. Ele permite memorizar o resultado (um valor) de uma função e reutilizá-lo sempre que as dependências dessa função não mudarem.

Quando se tem uma função que é executada repetidamente dentro de um componente, o uso do *useMemo* pode evitar a reexecução desnecessária dessa função, o que economiza tempo de CPU e melhora a responsividade da aplicação.

Ao utilizar *useMemo*, você especifica a função que deseja memorizar e uma matriz de dependências. Sempre que qualquer uma dessas dependências mudar, o *useMemo* recalcula o valor memorizado. Caso contrário, ele retorna o valor memorizado anteriormente, sem precisar executar a função novamente.

Por outro lado, deve-se ter cuidado com o uso excessivo desse hook, pois o consumo de memória pode se tornar bastante significativo tendo vários valores memorizados.

Sendo assim, o *useMemo* consiste numa ferramenta valiosa para melhorar o desempenho do React, evitando cálculos redundantes e a otimização da renderização de componentes

useRef

O hook *useRef*, ao contrário do *useState*, não provoca re-renderizações quando seu valor é atualizado, sendo ideal para armazenar dados que não causam

alterações visuais no componente.

Ele é particularmente útil para acessar diretamente elementos do DOM sem a necessidade de consultá-los repetidamente usando seletores ou métodos de busca.

Além disso, o *useRef* pode ser utilizado para manter valores mutáveis que não precisam acionar re-renderizações. Por exemplo, você pode utilizar um objeto current dentro do *useRef* para armazenar dados que precisam ser acessados ou atualizados por várias partes do componente sem causar re-renders desnecessários.

Outra aplicação comum do *useRef* é para interações com bibliotecas de terceiros que exigem acesso direto aos elementos do DOM. Ao criar uma referência com *useRef*, você pode passá-la para a biblioteca e manipular o elemento diretamente, se necessário.

Em resumo, o *useRef* é uma ferramenta poderosa no React para manter referências persistentes a elementos do DOM e valores mutáveis, sem causar re-renderizações desnecessárias. Ele é útil para acessar elementos do DOM, armazenar valores que não precisam ser re-renderizados e interagir com bibliotecas de terceiros de forma eficiente.

Referências Bibliográficas

Built-in React Hooks. **React Dev**, n.p. Disponível em: https://react.dev/reference/react/hooks. Acesso em 30 de mar. de 2024.

CALAZANS, Lucas. React.useMemo na prática. **DEV**, n.p., 2019. Disponível em: https://medium.com/reactbrasil/react-usememo-na-pr%C3%A1tica-692110771c01. Acesso em: 30 de mar. de 2024.

OLIVEIRA, Guilherme. Conhecendo o useRef do React. **DEV**, n.p., 2019. Disponível em: https://medium.com/@guigaoliveira_/conhecendo-o-useref-do-react-9d67e66. Acesso em: 30 de mar. de 2024.

React useCallback Hook. **W3Schools**, n.p. Disponível em: https://www.w3schools.com/react/react_usecallback.asp. Acesso em: 30 de mar. de 2024.v

React useMemo Hook. **W3Schools**, n.p. Disponível em: https://www.w3schools.com/react/react_usememo.asp. Acesso em: 30 de mar. de 2024.

React useReducer Hook. **W3Schools**, n.p. Disponível em: https://www.w3schools.com/react/react_usereducer.asp. Acesso em: 30 de mar. de 2024.

React useRef Hook. **W3Schools**, n.p. Disponível em: https://www.w3schools.com/react/react_useref.asp. Acesso em: 30 de mar. de 2024.

SELAIR, Guilherme. Hook useCallback - Entenda quando utilizá-lo? **DEV**, n.p., 2022. Disponível em: https://dev.to/guiselair/usecallback-hook-entenda-quando-utiliza-lo-3n3k. Acesso em: 30 de mar. de 2024.