SUMÁRIO

1. EFEITOS COLATERAIS COM USE EFFECT

O React Hook useEffect permite que um componente funcional execute **efeitos colaterais** quando o documento é carregado, e quando houverem alterações nos estados dos componentes.

De certa forma, o **useEffect pode ser compreendido como um "cache" de dados do componente**, que é ativado automaticamente sempre e apenas quando as dependências sofrem uma alteração. Por padrão, então, como dito, é ativado no carregamento da página, e após isso, após essas alterações. São esses os casos.

Adicionalmente, e fazendo alusão aos nomes que eram utilizados quando eram utilizados os componentes de classe, os dois casos, em que o componente é carregado, e em que é atualizado, podem ser descritos, respectivamente, como "**ComponentWillMount**" (componente será montado) e "**ComponentWillUptade**" (componente será atualizado).

1.1. A SINTAXE DO USE EFFECT

A sintaxe do useEffect é simples, primeiro ele é importado como qualquer outro Hook, e no escopo do componente, é implementado desta forma: useEffect(call-back, [dependências]). O primeiro parâmetro é uma função que executa os efeitos colaterais programados, e o segundo são as dependências, isto é, os dados selecionados que quando sofrem uma mudança disparam o useEffect.

O segundo argumento é opcional, mas imprescindível porque, na ausência da configuração das dependências, o useEffect é disparado uma única vez.

Veja um exemplo abaixo, em que há um state "Contador" definido como dependência do useEffect. Se é um efeito colateral, quando o valor do state for alterado com seu "setter", o useEffect será disparado imediatamente após.

```
const[Contador, setContador] = useState(0); //State Contador

useEffect( ()=>{ //Ativado automaticamente após a chamada de setContador

        console.log("O State 'Contador' foi alterado para:" + Contador)

}, [Contador])
```

Se a dependência “Contador” não fosse definida, esse bloco de código seria executado apenas na renderização do “App”.

2. LIMPEZA DE EFEITOS COLATERAIS E COMPONENTS LIFE CICLE

Existem dois tipos de efeitos colaterais possíveis que podem ser criados com useEffect: os que ocorrem e não devem ser “limpos”, e o que devem ser porque deixam para trás “rastros indesejáveis”.

2.1. PROCESSOS ENVOLVENDO COMPONENTES

Para ficar claro, um componente renderizado é um componente montado. O nome do processo de montar um componente é “**ComponentWillMount**”, e ele sempre dispara, por padrão, o useEffect. Ou seja, **a montagem de componentes sempre produz efeitos colaterais**, isto é, sempre ativa um useEffect.

O segundo tipo de processo, de atualização de componente, “**ComponentWillUpdate**”, **pode disparar hooks useEffect, ou não**. Ou seja, a atualização de componentes pode ou não estar programada para gerar um efeito colateral.

O terceiro tipo de processo é o “**ComponentWillUnmount**”, que é basicamente é a retirada de um componente previamente renderizado. O desfazer de sua renderização. **É neste caso que a limpeza dos efeitos colaterais pode ser necessária**.

2.2. LIMPEZA DOS EFEITOS COLATERAIS

Quando um componente é montado, alterado, e lança um efeito colateral na aplicação, o que deve ocorrer quando é desmontado? Se o seu efeito colateral for, planejadamente, um que não necessita da sua presença após lançado, não existem problemas.

A limpeza é o processo de, na desmontagem do componente, retirar também os efeitos colaterais que produziu e que só faziam sentido na sua presença. Na maioria das vezes, isso é necessário quando os efeitos colaterais implicam em alterações fora do pequeno escopo de atuação do componente montado.

Por exemplo, na renderização de um componente de modal, digamos que o estilo de outro componente X é alterado, como efeito colateral. O que deve ocorrer se o modal for fechado? O fechamento do modal, que deve ser entendido como a desmontagem do componente de modal, deve também ser seguido da reversão da alteração de estilo do componente X. **Isto seria o ato de limpar os rastros de um processo de “component unmount”**.

2.2.1. COMO REALIZAR A LIMPEZA

A limpeza não é feita com uma função nativa, ou com métodos prontos, mas com o retorno de uma função programada no useEffect do componente que poderá vir a ser desmontado, assim:

```
useEfffect( ()=>{
        //Programação do useEffect
        return () => {
        //Programação da limpeza
        }
}, [dependencies]);
```