Nome: Kefflen Moreno Ramos

Diferença entre usar hooks ou componente de classe

React hooks foram adicionados a partir da versão 16.8, e oferecem muitas funcionalidades que até esta versão seria necessário implementar um componente com classe.

Os componentes reacts tem um ciclo de vida na aplicação. Quando elas que seriam quando elas são renderizadas, quando seu estado muda, ou quando elas são desmontadas.

Gerenciamento de estado

Componente de classe

Para gerenciar um estado de um componente de classe é necessário usar o método **"setState"**, que recebera um objeto que com os atributos/estado que serão alterados e seus valores. Os estados do componente de classe ficam no atributo state, que é um objeto contendo todos os estados

Usando hooks

Para ter variáveis que representam o estado do componente deve ser usado o hook **"useState"**, que recebe como parâmetro o estado inicial ou uma função que será executado apenas no ciclo "mount", que definira o estado inicial. Este hook retorna um array contendo dois itens, sendo o primeiro o próprio estado e o segundo uma função que deve ser usada para modificar o estado, provocando a renderização do componente.

Tambem existe um hooks chamado **"useRef"**, que funciona quase igual ao useState, mas retorna uma referência que guarda o valor no atributo "current", que pode ser alterado sem provocar um renderização do componente

Ciclo de vida

1. ### Mount

   Componente com classe:
   Para implementar esse ciclo de vida em um componente com classe a necessário implementar um método chamado **"componenteDidMount"** onde a gente implementa o algoritmo que deve ser executado no momento de renderizar pela primeira vez.
   Usando hooks:
   Para implementar esse ciclo usando hooks é necessário usar o Hook **'useEffect'** passando uma call-back, onde é implementado o algoritmo que deve ser executado, e uma lista vazia para demonstrar que esse useEffect só será executado no primeiro momento que for renderizado

2. Update

Componente com classe

Para implementar esse ciclo de vida é necessário implementar o método **“componenteDidUpdate”**, que recebe como parâmetros o props e o estado anterior ao update, além de um snapshot, se tiver implementado o método “getSnapshotBefoteUpdate”.

Usando hooks

A gente pode usar o **“useEffect”** passando uma call-back, deperecias que seria um array contendo os estados do componente que quando alterados irá provocar a execução da call-back. Diferente do componente de classe essa sintaxe oferece mais controle deste ciclo de vida podendo passar comportamentos diferentes dependendo do estado que foi alterado

3. Umount

Componente com classe

O método responsável por este ciclo de vida é o **“componentWillUmount”**, será chamado quando o componente será defeito e destruído

Com hooks

Para ter esse mesmo comportamento a gente pode usar o **“useEffect”**, passando uma função que irá retornar uma função que deve ser executado no momento que irá ocorrer o “umount”