Efeitos Colaterais e useEffect

Labenu_

O que vamos ver hoje?

- O que são efeitos colaterais (side effects)
- Hook useEffect()
- Ciclos de vida com o useEffect()

Efeitos Colaterais

Labenu_

O que são Efeitos Colaterais? 🏌️

- **Exemplo da vida real:** Trocar de casaco

  - Ação independente
  - Consequência global
  - Gera efeito colateral

O que são Efeitos Colaterais? 

- Um Efeito Colateral (ou *Side Effect*) é quando uma função altera algo **fora do seu próprio escopo**

- Em componentes de classe, fazemos isso usando **métodos de ciclo de vida**
  - `componentDidMount()`
  - `componentDidUpdate()`

Exemplo de componentDidMount()

- Quando o componente renderizar pela primeira vez, faça uma requisição GET para buscar todos os produtos

  - O efeito colateral é a busca por todos os produtos

Exemplo de componentDidUpdate()

- Temos um estado chamado “idSelecionado”
- Sempre que o componente renderizar e for identificada a mudança nesse estado, faça uma requisição GET para buscar um produto por id
- O efeito colateral é a busca de um produto por id

useEffect()

Labenu_

useEffect - Principais conceitos 🧦

- Hook que implementa o ***USO*** de efeitos colaterais (side ***EFFECT***)

- É disparado sempre **depois de uma renderização**

- Pode ser chamado **mais de uma vez** no componente

useEffect - Import 🧦

- Ele deve ser importado do React entre chaves

```
import React, {useEffect} from 'react'
```

Adicionar isso ao topo do arquivo quando quisermos usar o useEffect()

useEffect - Argumentos🧦

- Recebe dois argumentos:

  1) Arrow function = o que fazer

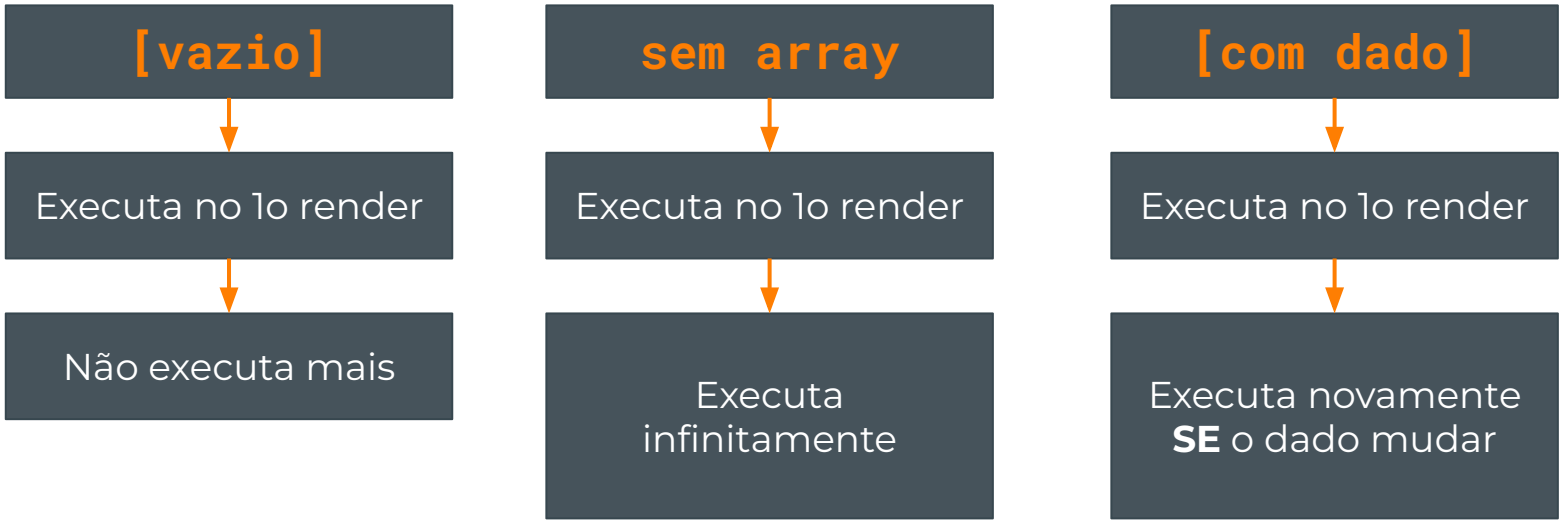
  2) Array = quando fazer

```
useEffect(( ) => { O que fazer }, [ Quando fazer ])
```

useEffect - Exemplo 🧦

```
useEffect(( ) => {
  axios.get("https://www.minha-api.com/v1/produtos")
       .then((res) => {
         console.log(res.data)
       })
       .catch((err) => {
         console.error("Erro ao buscar produtos")
       })
}, [ ])
```

useEffect - Array de dependências 🧦



Array de Dependências - Vazio [ ] 🧦

- A função será executada **uma única vez**, depois da primeira renderização do componente

- Um caso de uso comum é quando queremos carregar dados de APIs na tela assim que ela renderizar

Array de Dependências - [dados] 🧦

- A função será executada logo após a **primeira renderização do componente**, a diferença agora é sua execução em novas renderizações, desde que os dados tenham sido alterados

- Um caso de uso é a sincronização de um elemento controlado na tela com um outro que possui as informações vindas de uma API

Array de Dependências - sem array 🧦

- A função será executada sempre depois de uma renderização

- Evitamos utilizá-la para não gerar loops infinitos e outros erros imprevisíveis

Exemplo assistido

- Crie um componente funcional que simula um contador

- Utilize o useEffect, testando as três maneiras apresentadas

- A função de efeito colateral deve disparar um console.log(“O efeito colateral foi executado”)

Exercício Parte 1 - Enunciado

Exemplo de Substituição do componentDidMount

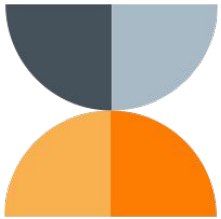
- Crie um componente funcional que lista os nomes de personagens de Star Wars

- Link do endpoint:

GET https://swapi.py4e.com/api/people/?page=1

Exercício Parte 1 - Dicas

- O comportamento será idêntico ao componentDidMount
- O array de dependências será vazio [ ]
- A função de efeito colateral irá utilizar o axios para fazer uma requisição GET e salvará o resultado no estado
- Teremos um map do estado, retornando cada nome da lista envolto por alguma tag HTML



Pausa para relaxar 😴

10 min

- Efeitos colaterais são funções que afetam algo **fora do seu escopo**
- No React, efeitos colaterais acontecem **pós-renderização**
- Em componentes funcionais, usamos o hook **useEffect()**

Exercício Parte 2

Exemplo de Substituição do componentDidUpdate

- Agora crie uma funcionalidade que permita selecionar as personagens por nome e mostre seu **ano de nascimento**
- Link do endpoint:

GET https://swapi.py4e.com/api/people/:index

path param index = número iniciando do 1, de acordo com a ordem da personagem na lista do endpoint anterior

Erros comuns

Labenu_

Erros comuns 🧦⚠️

- Esquecer de colocar a dependência no array (funciona uma vez, depois para de funcionar)

- Chamar a função de atualização da própria dependência dentro da função de efeito colateral (loop infinito)

Erros comuns 🧦⚠️

- Esquecer que é possível chamar mais de um useEffect no mesmo componente

- Colocar async na função de efeito colateral (bugs inesperados de fluxo)

Resumo

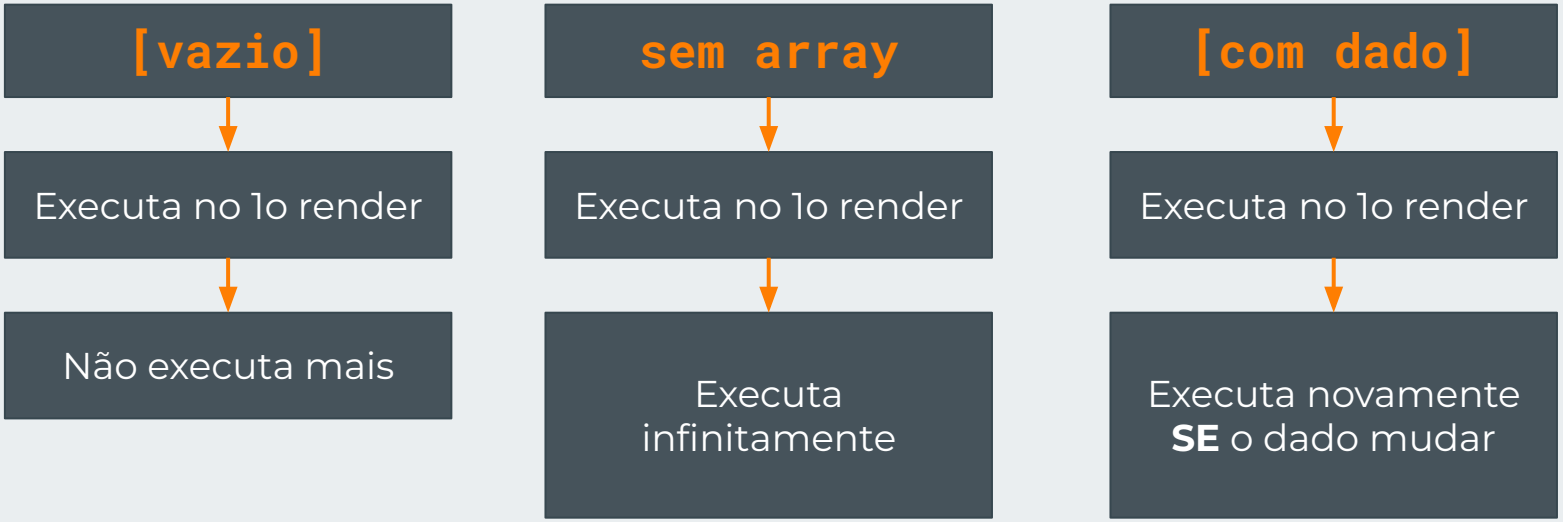
Labenu_

Resumo 📓

- Um Efeito Colateral (ou *Side Effect*) é quando uma função altera algo **fora do seu próprio escopo**
- useEffect é o Hook que implementa o ***USO*** de efeitos colaterais (side ***EFFECT***)
- É disparado sempre **depois de uma renderização**

Resumo 📓

Array de dependências



Dúvidas? 🧐

Labenu_

Obrigado(a)!