AULA useEffect()

useEffect()

O useEffect é um Hook do React que permite que você execute "efeitos colaterais" em componentes de função.

Efeitos colaterais podem ser funções que são executadas após o acontecimento de algo, por exemplo: carregamento total da página, alteração de uma variável, etc.

useEffect()

useEffect(() => { ... }, []):

A sintaxe do useEffect é similar a de uma função, mas ao final há um parâmetro.

Esse parâmetro pode indicar que deve ser executado toda vez que o componente atualizar, para isso, passa-se um array vazio [] .

Ou pode-se passar uma variável e então ele só será executado se o valor dessa variavel mudar.

useEffect()

useEffect(() => { SEU CODIGO AQUI }, []):

VARIAÇÕES

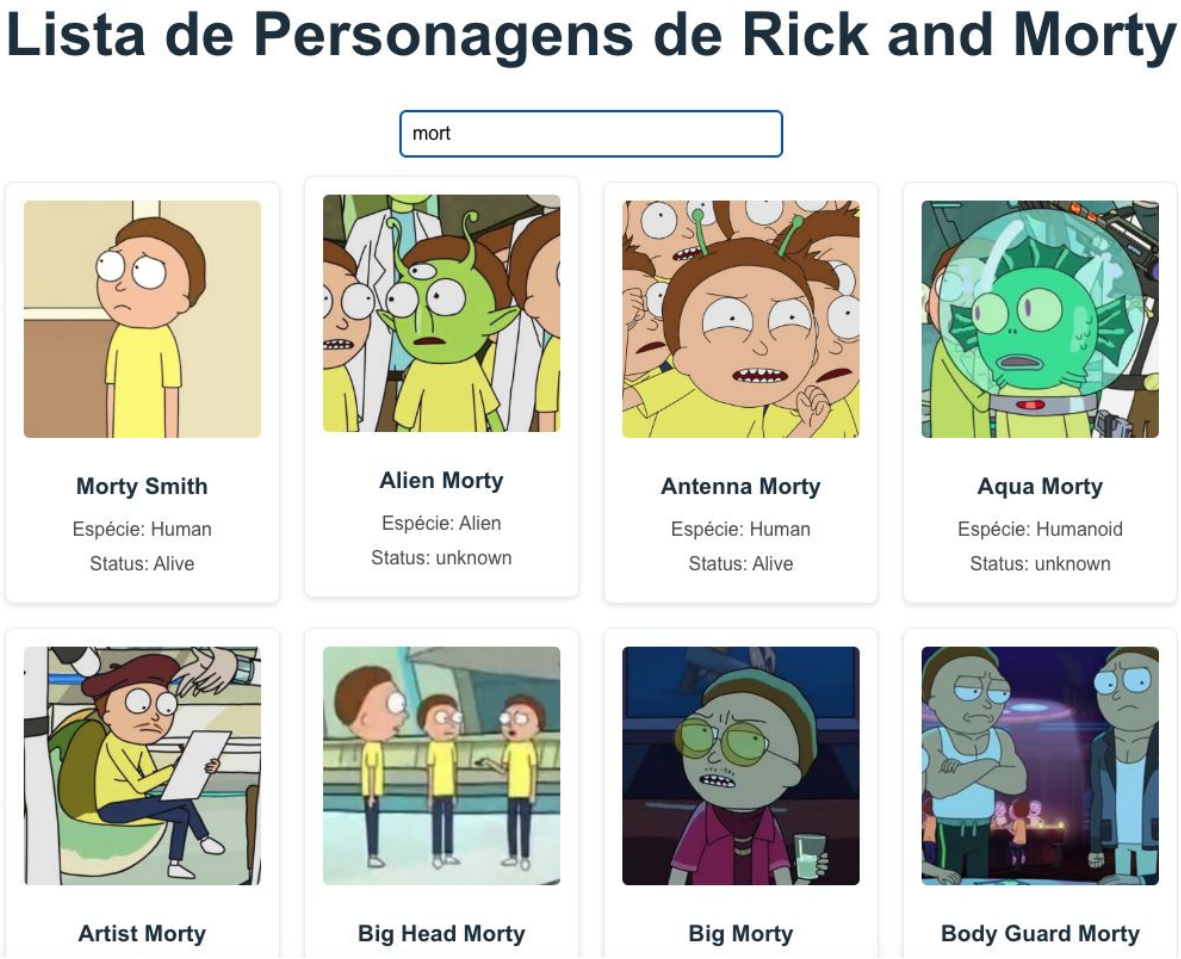
1. sem parametro, executa toda vez que o componente / tela for atualizado.
2. Passando uma variavel, a função useEffect só será executada quando essa variável alterar seu valor.
3. Com array vazio, executa somente 1x, é importante quando se deseja fazer o download de dados, somente 1x.

```jsx
import React, { useState, useEffect } from 'react';

export default function MensagemBoasVindas() {
  const [userName, setUserName] = useState('Visitante'); // Estado para o nome do usuário
  const [cliques, setCliques] = useState(0); // Um estado extra para mostrar que o componente pode renderizar por outros motivos

  // Efeito: Registra uma mensagem no console quando o 'userName' muda
  useEffect(() => {
    console.log(`Olá, ${userName}! Bem-vindo(a) de volta!`);
    // Imagine que aqui poderia ter uma chamada para uma API de analytics,
    // registrando a entrada de um novo usuário ou a mudança de usuário logado.
  }, [userName]); // <<-- O EFEITO SÓ RODA SE userName MUDAR!

  return (
    <div>
      <h1>Componente de Boas-Vindas</h1>
      <p>O nome de usuário atual é: **{userName}**</p>
      <p>Cliques: {cliques}</p>

      <button onClick={() => setUserName('Alice')}>
        Definir Usuário para Alice
      </button>
      <button onClick={() => setUserName('Bob')}>
        Definir Usuário para Bob
      </button>
      <button onClick={() => setUserName('Visitante')}>
        Definir Usuário para Visitante
      </button>

      <button onClick={() => setCliques(cliques + 1)}>
        Incrementar Cliques (Não muda o usuário)
      </button>
    </div>
  );
}
```

Variação 2

useEffect()

Vamos criar um componente React que lista os personagens do desenho Rick & Morty a partir dos dados de uma API pública. Além disso, iremos colocar um campo de busca para filtrar os personagens pelo nome. Tudo isso será feito com o useState() que já conhecemos, o fetch() e o useEffect().



```jsx
import { useState, useEffect } from 'react';
import './ListaPersonagens.css'; // 1. Importa o arquivo CSS

export default function ListaPersonagens() {
 const [personagens, setPersonagens] = useState([]);
 const [searchTerm, setSearchTerm] = useState('');


 return (
   <div className="container">
     <h1>Lista de Personagens de Rick and Morty</h1>
     <input
       type="text"
       className="search-input"
       placeholder="Buscar personagem por nome..."
       value={searchTerm}
       onChange={e => setSearchTerm(e.target.value)}
     />
     <div className="character-grid">
       {personagens.map(personagem => (
         <div key={personagem.id} className="character-card">
           <img src={personagem.image} alt={personagem.name} />
           <h3>{personagem.name}</h3>
           <p>Espécie: {personagem.species}</p>
           <p>Status: {personagem.status}</p>
         </div>
       ))}
     </div>
   </div>
 );
}
```

Parte 1: Criação do componente e o que será exibido ao usuário.

Variação 3

```
useEffect(() => {
  const fetchPersonagens = async () => {

    const apiUrl = `https://rickandmortyapi.com/api/character/?name=${searchTerm}`;

    try {
      const response = await fetch(apiUrl);
      if (!response.ok) {
        // A API do Rick and Morty retorna 404 quando não encontra um personagem,
        // o que é um resultado esperado, não um erro de aplicação.
        if (response.status === 404) {
          setPersonagens([]);
        } else {
          throw new Error(`Erro HTTP: ${response.status}`);
        }
      } else {
        const data = await response.json();
        setPersonagens(data.results || []);
      }
    } catch (err) {
      console.error('Erro ao buscar personagens:', err);
    }
  };

  fetchPersonagens();

}, [searchTerm]); // O efeito re-executa quando 'searchTerm' muda
```

Parte 2: Adição do useEffect. Antes do return da função, adicione.

Variação 3

```css
/* Estilos para o container principal */
.container {
 padding: 20px;
 font-family: Arial, sans-serif;
}

/* Estilo para o campo de busca */
.search-input {
 padding: 10px;
 width: 300px;
 margin-bottom: 20px;
 border-radius: 5px;
 border: 1px solid #ccc;
 font-size: 1rem;
}

/* Mensagem de erro */
.error-message {
 color: red;
 font-weight: bold;
}

/* Grid que alinha os cards dos personagens */
.character-grid {
 display: grid;
 grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
 gap: 20px;
}
```

Parte 3: Link o CSS.

Variação 3

```css
/* Estilo individual para cada card de personagem */
.character-card {
 border: 1px solid #eee;
 border-radius: 8px;
 padding: 15px;
 text-align: center;
 box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
 transition: transform 0.2s ease-in-out;
}

.character-card:hover {
 transform: translateY(-5px);
}

/* Imagem do personagem dentro do card */
.character-card img {
 width: 100%;
 height: auto;
 border-radius: 5px;
 margin-bottom: 10px;
}

.character-card h3 {
 margin: 10px 0;
 font-size: 1.2rem;
}

.character-card p {
 margin: 5px 0;
 color: #555;
}
```

Parte 3: Link o CSS.

Variação 3