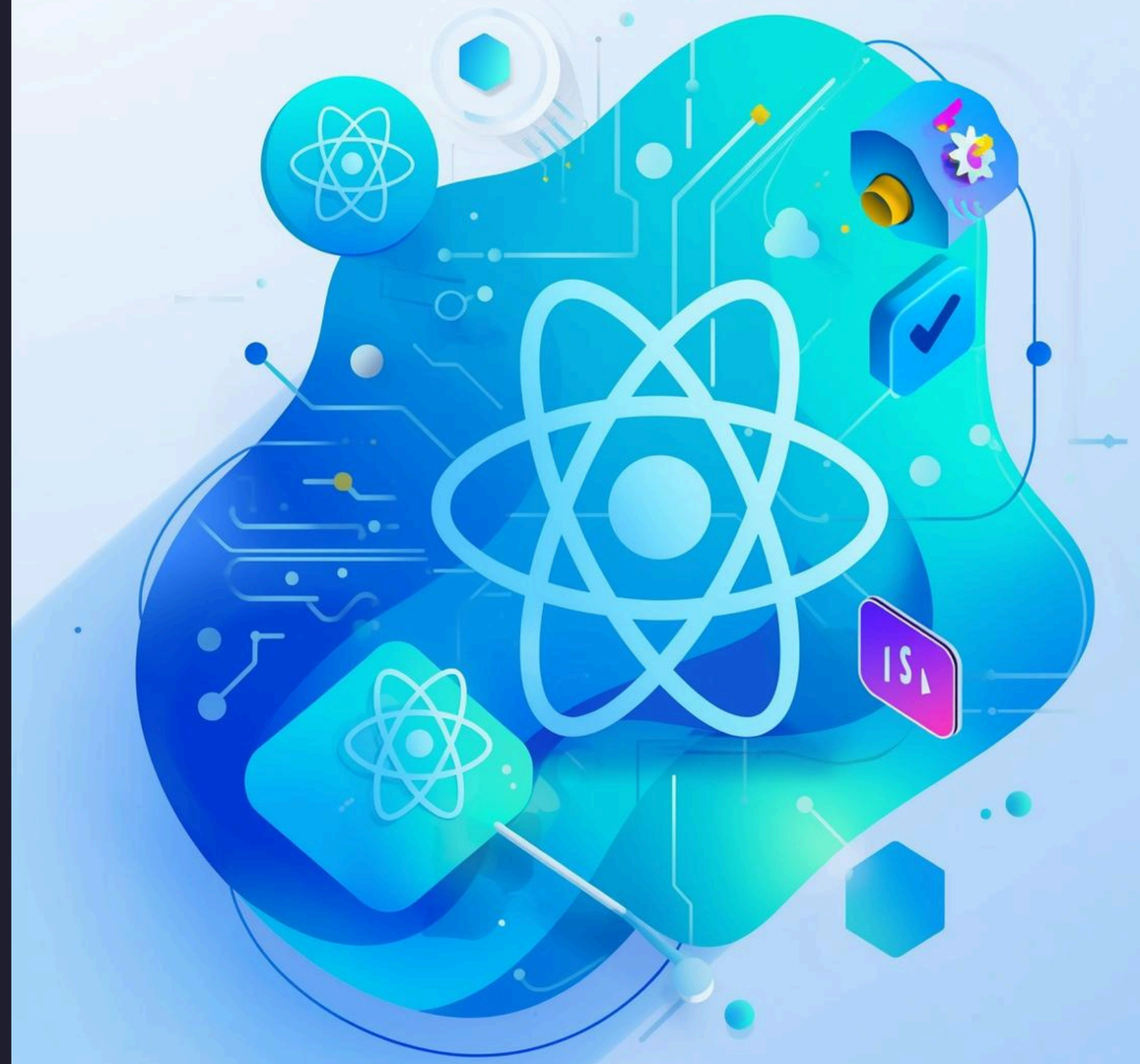
AULA 03

Hooks

useEffect e Ciclo de Vida



Profª Josiane Lopes

BOLSA FUTURO DIGITAL | PROGRAMA DE FORMAÇÃO ACELERADA
INSTITUTO FEDERAL Sergipe
cepedi
Softex
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

O que são Hooks?

Funções especiais que permitem:

Gerenciar estado

Acessar ciclo de vida

Controlar side effects

Eexemplos principais:

useState
useEffect
useContext
useMemo
useCallback

OBS: Hooks só funcionam em componentes funcionais.

Side Effects (Efeito Colateral)

Importância na aplicação

Side effects → fundamentais para a interação da aplicação. São mudanças no estado do Sistema ou Dados fora do escopo esperado da função.

Introdução ao useEffect

useEffect = executar código quando algo acontece no componente.

- Faz parte do ciclo de vida
- Ideal para:
  - Buscar dados (fetch)
  - Manipular DOM
  - Timers (setInterval, setTimeout)
  - Conexão com APIs externas
  - Logs e debugs

Introdução ao useEffect

Comparação com componentes de classe:

- componentDidMount
- componentDidUpdate
- componentWillUnmount

useEffect

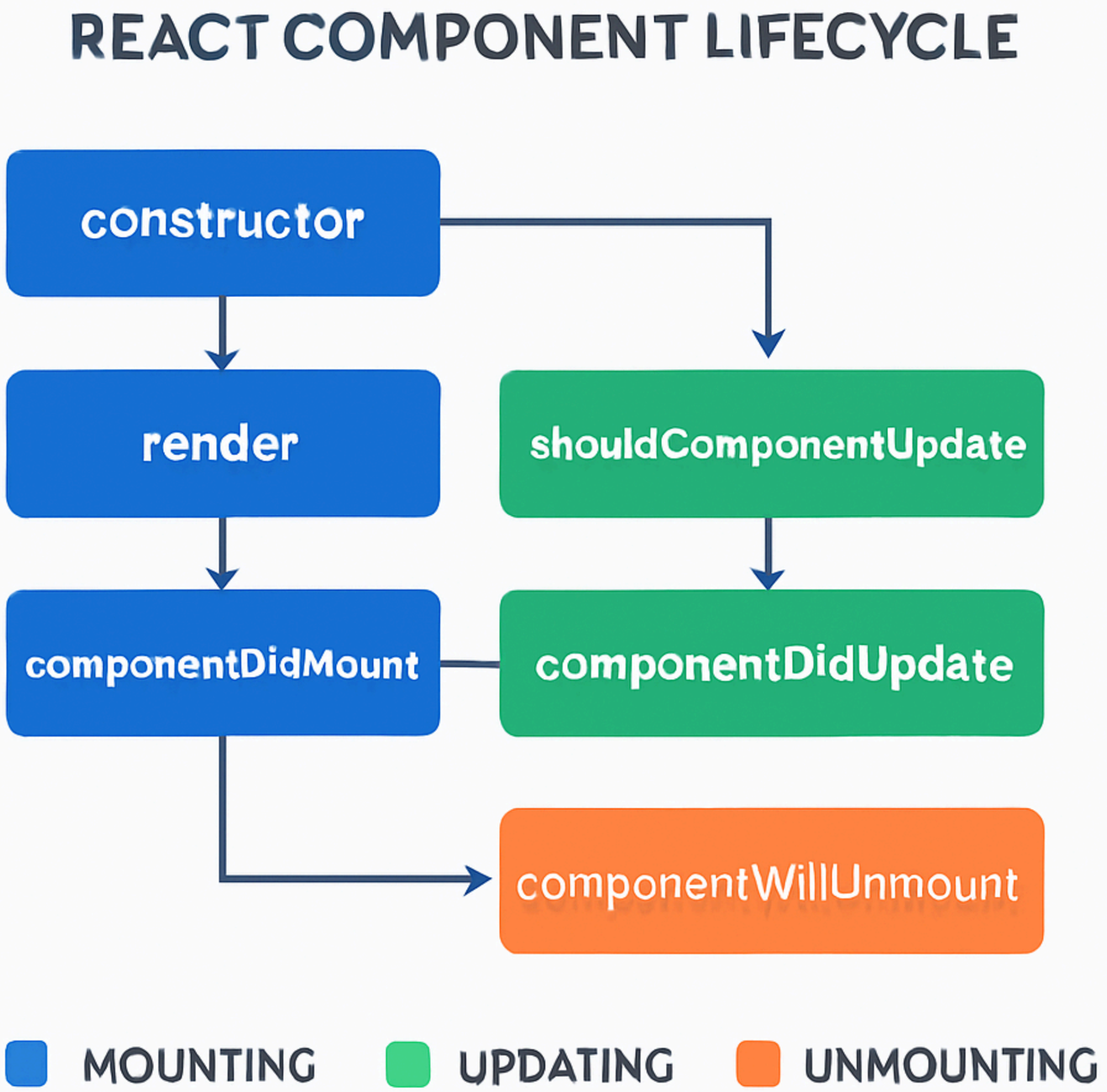
SINTAXE BÁSICA

```
useEffect(() => {
  // código executado após a renderização
});
```

Ciclo de Vida

Componentes Funcionais com useEffect:

1. Renderiza o componente;
2. Executa a função do useEffect
3. Atualização do state → re-renderiza
4. useEffect roda novamente
5. Se houver cleanup → executa antes do próximo efeito



useEffect quando o componente monta

```
useEffect(() => {
  console.log("Componente montou!");
}, []);
```

Dependência vazia → roda apenas uma vez

Equivalente a componentDidMount

useEffect quando um estado muda

```
useEffect(() => {
  console.log("O nome mudou:", nome);
}, [nome]);
```

Executa somente quando o valor da Dependência se altera

Exemplo prático

Contador que registra no título da página

```jsx
import { useState, useEffect } from "react";

function Contador() {
  const [cliques, setCliques] = useState(0);

  useEffect(() => {
    document.title = `Cliques: ${cliques}`;
  }, [cliques]);

  return (
    <button onClick={() => setCliques(cliques + 1)}>
      Clique {cliques}
    </button>
  );
}
```

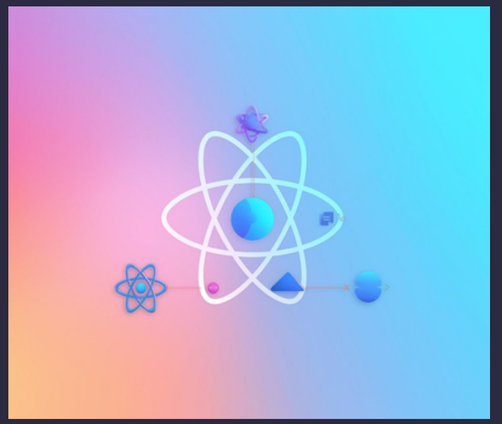
Cleanup (limpeza)

Usado para:

- **Cancelar timers**
- **Limpar listeners**
- **Cancelar requisições**

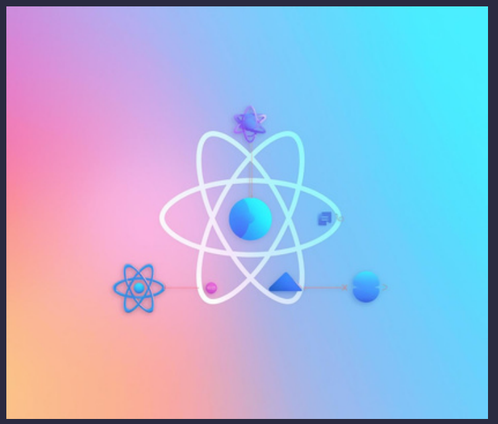
```
useEffect(() => {
  const id = setInterval(() => {
    console.log("rodando...");
  }, 1000);

  return () => clearInterval(id);  // cleanup
}, []);
```

Boas Práticas



- ✓ Utilize dependências corretamente
- ✓ Coloque variáveis usadas no effect dentro do array
- ✓ Não execute efeitos dentro de loops
- ✓ Divida useEffects diferentes por responsabilidade
- ✓ Cuidado com efeitos que disparam infinitamente
- ✓ Não deixe efeitos fazendo trabalho pesado

Erros Comuns



❌ Esquecer dependência

❌ Colocar função anônima dentro do JSX que dispara renderizações extras

❌ Atualizar estado dentro do effect sem condicional

❌ Fazer fetch sem cleanup (efeito zumbi)

Mini-Demonstração

Carregando dados de uma API fake

```
useEffect(() => {
  fetch("https://jsonplaceholder.typicode.com/posts/1")
    .then(res => res.json())
    .then(data => setPost(data));
}, []);
```

Mini-Atividades

Prática

Mini-atividade 1

Crie um componente Relógio que mostra a hora atual e atualiza a cada 1 segundo usando useEffect + setInterval.

Mini-atividade 2

Crie um componente que:

- tenha uma caixa de texto
- sempre que o usuário digitar, salve no estado
- useEffect mostra no console:
- "O usuário digitou: xxxxx"

Desafios

Prática

Desafio 1

Criar um contador com:

- botão aumentar
- botão diminuir
- useEffect exibindo:
  - "Valor mudou para: X"

Desafio 2

Criar um componente Requisicao que:

- ao montar, faça fetch em uma API
- mostre "Carregando..." enquanto busca
- exiba o retorno depois
- trate erro com uma mensagem amigável

Desafio 3

Criar um componente com:

- tema claro/escuro
- state para tema
- useEffect altera o document.body.style .background

Obrigada!

Profª Josiane Lopes