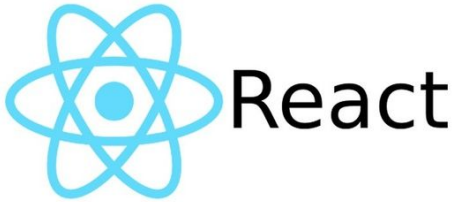
centro universitário FEI

CCP150
DES. DE APLICATIVOS MÓVEIS

Prof. Isaac
isaacjesus@fei.edu.br

Ciclo de Vida de Componentes (lifecycle)

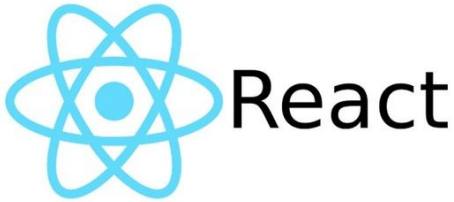
Ciclo de Vida de Componentes



- Cada componente do React passa pelas seguintes fases:
  - **Montagem *(mounting)***
  - **Atualização *(updating)***
  - **Desmontagem *(unmounting)***

Montagem significa colocar os elementos no DOM.

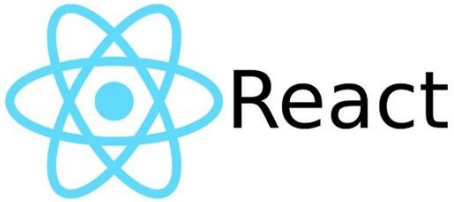
Ciclo de Vida de Componentes



- **Montagem:** Componente sendo montado no DOM do navegador.
  - Métodos que são chamados nessa ordem:
    - ***constructor:*** primeiro método executado.
    - ***getDerivedStateFromProps***: Método chamado antes de renderizar os elementos.
    - ***render*:** monta o componente no navegador.
    - ***componentDidMount:*** é executado depois que o componente foi montado no DOM.

https://react.dev/reference/react/Component

Montagem

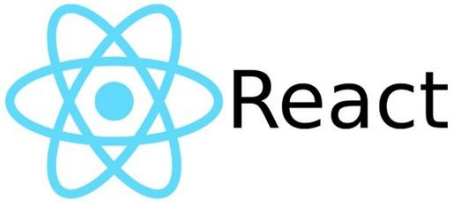


getDerivedStateFromProps(): Com este método é possível atualizar o state com os valores do props. Ele recebe o state e o props como argumentos e retorna o objeto state atualizado.

Exemplo:

https://codepen.io/Isaac-Jesus-Silva/pen/LYvJYBb

Montagem

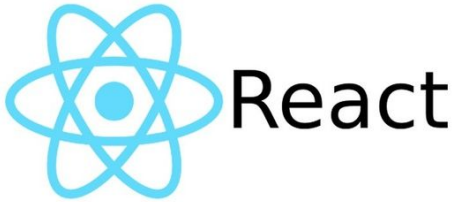


componentDidMount(): Executado depois que o componente foi montado no DOM. É aqui que você executa instruções que exigem que o componente já esteja no DOM.

Exemplo:

https://codepen.io/Isaac-Jesus-Silva/pen/JjVajQE

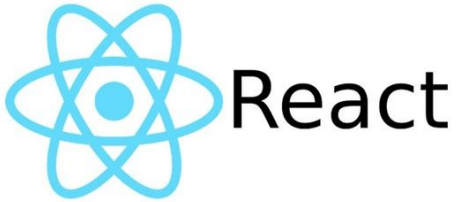
Ciclo de Vida de Componentes



- **Atualização:** O componente pode ser atualizado pelo envio de novas ***props*** ou pela atualização do seu ***state***
  - 5 métodos que são chamados nessa ordem:
    - ***getDerivedStateFromProps()***
    - ***shouldComponentUpdate()***
    - ***render()***
    - ***getSnapshotBeforeUpdate()***
    - ***componentDidUpdate()***

https://react.dev/reference/react/Component

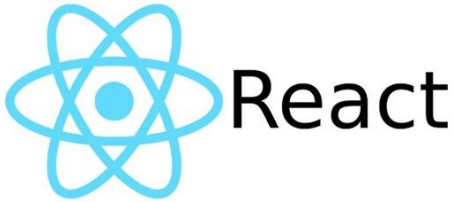
Ciclo de Vida de Componentes



- **Atualização:**
  - ***getDerivedStateFromProps():*** Primeiro método executado na atualização.
  - ***shouldComponentUpdate***: Nesse método você pode retornar um valor booleano que especifica se o React deve continuar com a renderização ou não. O valor padrão é **true**.
  - ***getSnapshotBeforeUpdate:*** Esse método tem acesso aos valores do props and state antes da atualização, combinado com o componentDidUpdate, é possível confirmar se os valores foram atualizados.
  - ***componentDidUpdate:*** É executado depois que o componente é atualizado no DOM.

https://react.dev/reference/react/Component

Montagem

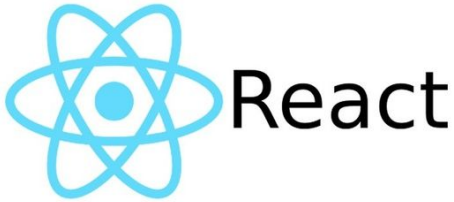


shouldComponentUpdate(): Nesse método você pode retornar um valor booleano que especifica se o React deve continuar com a renderização ou não. O valor padrão é **true**.

Exemplo:

https://codepen.io/Isaac-Jesus-Silva/pen/yLrxyEa

Montagem

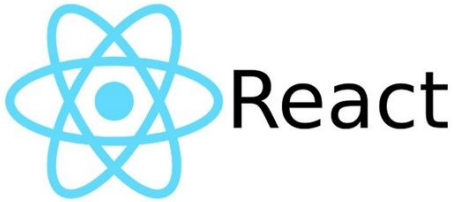


componentDidUpdate(): Executado depois que o componente foi atualizado no DOM. É aqui que você executa instruções que exigem que o componente já esteja no DOM.

Exemplo:

https://codepen.io/Isaac-Jesus-Silva/pen/jORvELE

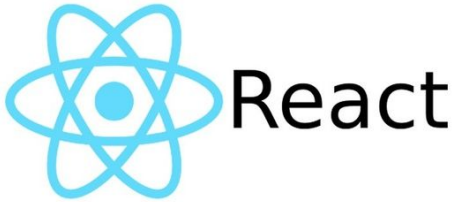
Ciclo de Vida de Componentes



- **Desmontagem:** O componente não é mais necessário e será desmontado do DOM
    - Métodos:
        - ***componentWillUnmount:*** último método no ciclo de vida. Executado imediatamente antes de o componente ser removido do DOM

https://react.dev/reference/react/Component

Montagem

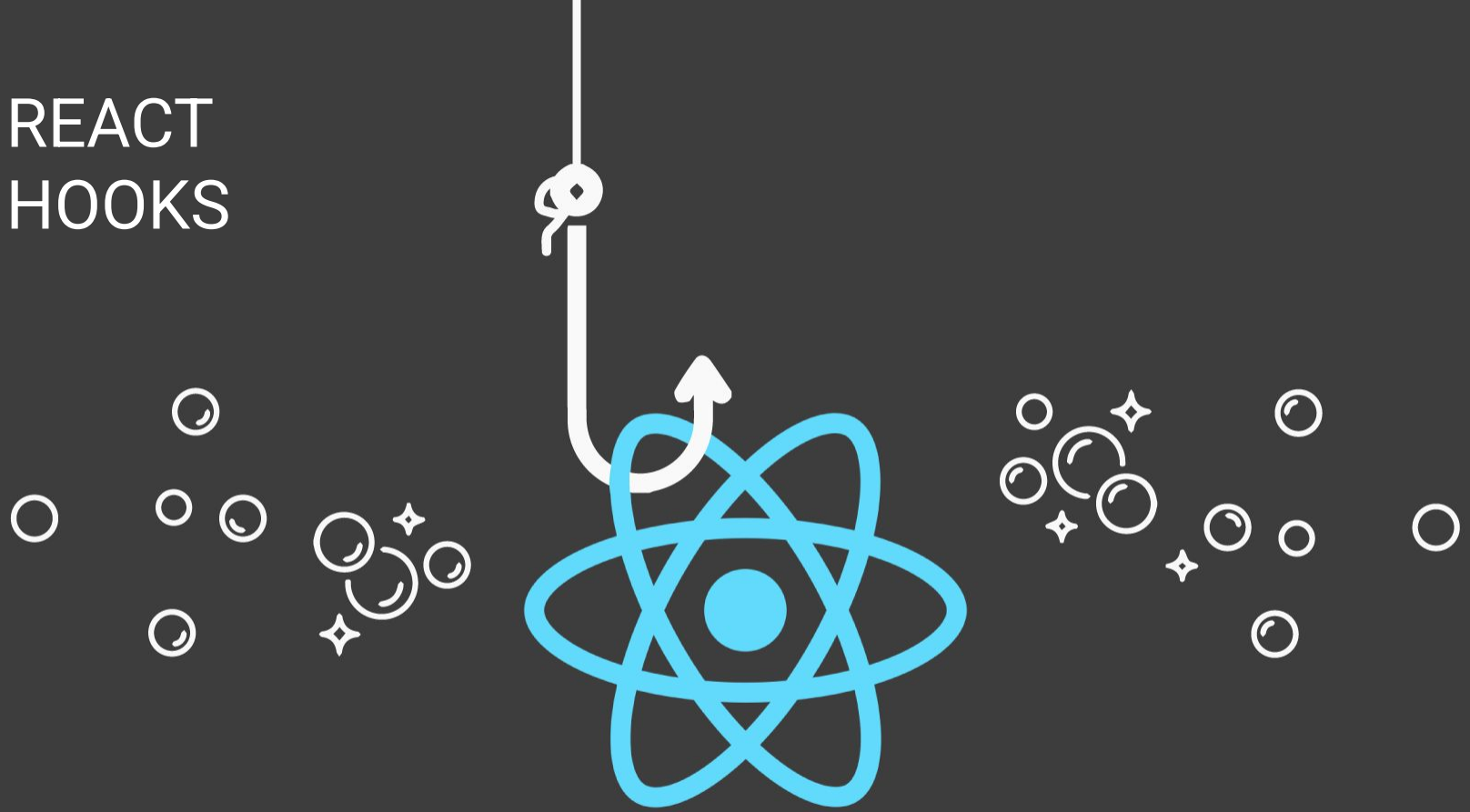


componentWillUnmount():

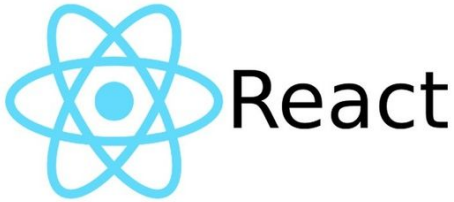
Exemplo:

https://codepen.io/Isaac-Jesus-Silva/pen/yLrxNLb

REACT HOOKS

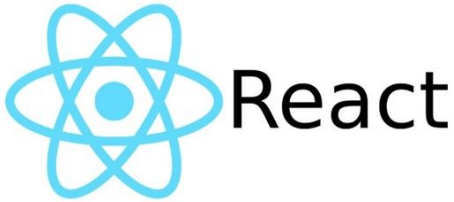


Hooks



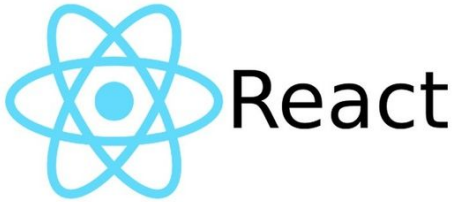
- Os Hooks adicionam ao React a possibilidade de usarmos o **state** sem a necessidade de criar classes
- Até o surgimento dos Hooks (2019), os **states** eram informações que existiam somente nos **Componentes de Classe**
- Os Hooks permitem que utilizemos os estados **(states)** em **Componentes de Função (Componentes Funcionais)**

Hooks



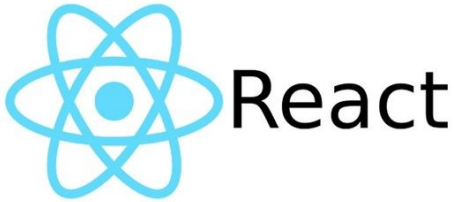
- Os desenvolvedores do React perceberam que o uso de classes podia deixar o código mais difícil, principalmente por conta da preocupação com o **this**
- Os Hooks não alteram o funcionamento do React. Tudo que já funcionava continua funcionando
- A ideia foi acrescentar outra possibilidade de desenvolvimento

Hooks



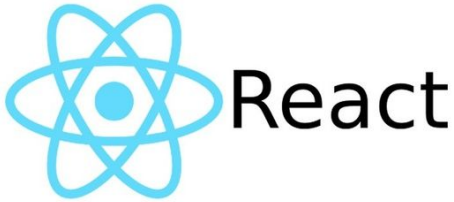
- **Hook de Estado (State Hook)**
  - **states** estavam vinculados às classes, pois precisamos do método **setState** e do construtor para criá-los
  - O primeiro Hook que vamos ver é o **useState**
    - **import { useState } from ‘react’;**
  - Com o **useState** vamos conseguir manipular o **state** dentro de um **Componente de Função**

Hooks



- **Hook de Estado (State Hook)**
  - **useState** é utilizado da seguinte forma:
    - **const [variavel, setVariavel] = useState(valor);**

Exemplo

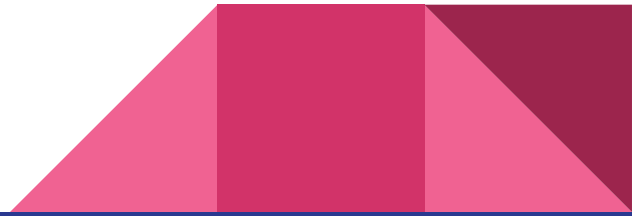


https://codepen.io/Isaac-Jesus-Silva/pen/BaExXgp

https://codepen.io/Isaac-Jesus-Silva/pen/dyLKbNY

Exemplo

- Desenvolva um app para incrementar ou decrementar um valor quando os respectivos botões forem pressionados, utilizando Componentes de Função



Hooks

- **Hook de Efeito (Effect Hook)**
  - **useEffect :** podemos pensar no **useEffect** com funcionamento similar ao ***componentDidMount***, ***componentDidUpdate***, e ***componentWillUnmount*** combinados.
    - **import React, { useState, useEffect } from 'react';**

https://react.dev/reference/react/useEffect

Hooks

- **Hook de Efeito (Effect Hook)**
  - **useEffect :** podemos aceita dois argumentos, o segundo é opcional.

    - **useEffect(<function>, <dependency>);**

https://react.dev/reference/react/useEffect

Hooks

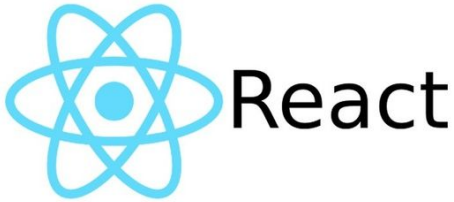
- **useEffect :** Usando com o argumento <dependency> sendo [ ].

  Equivalente a **componentDidMount**

```
useEffect(() => {
    //Executa somente na primeira renderização
}, [ ] );
```

https://codepen.io/Isaac-Jesus-Silva/pen/PogdPYm

Exercício



Reescreva o código abaixo usando apenas componente de função com useState e useEffect.

https://codepen.io/Isaac-Jesus-Silva/pen/JjVajQE

Hooks

- **useEffect :** Usando sem o argumento <dependency>.

  Equivalente a **componentDidUpdate**

```
useEffect(() => {
    //Executa a cada renderização
});
```

https://codepen.io/Isaac-Jesus-Silva/pen/LYvJpVV

Hooks

- **useEffect :** Usando com o argumento <dependency> com [props, state].

Equivalente a **componentDidUpdate** mas somente para o **state** ou **props** informado.

```
useEffect(() => {
    //Executa na primeira renderização
    //Executa toda vez que o props e o state são atualizados
}, [ props, state ] );
```

https://codepen.io/Isaac-Jesus-Silva/pen/BaEOoLm

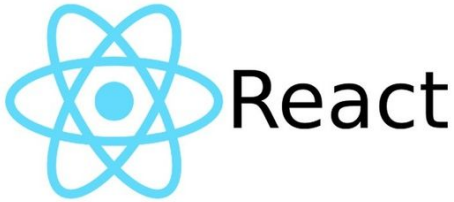
Hooks

- **useEffect :** Equivalente a **componentWillUnmount**

```
useEffect(() => {
        return() => {
                //Será executado quando o componente for removido.
        }
}, [ ] );
```

https://codepen.io/Isaac-Jesus-Silva/pen/oNOPbdd

Exercício



Crie uma aplicação React Web com **Hooks** para um **cronômetro**. O objetivo é que o usuário possa iniciar, pausar e zerar a contagem do tempo, com a exibição em tempo real.

Requisitos:

- Controle de Tempo: O cronômetro deve ter 3 botões: Iniciar, Pausar e Zerar a contagem.
- Formato de Exibição: O tempo deve ser exibido em um formato legível, como **HH:MM:SS** (horas, minutos, segundos).