

# Lifecycle Methods

- › Ciclo de vida dos componentes
  - › Componentes de Class
  - › Componentes Funcionais

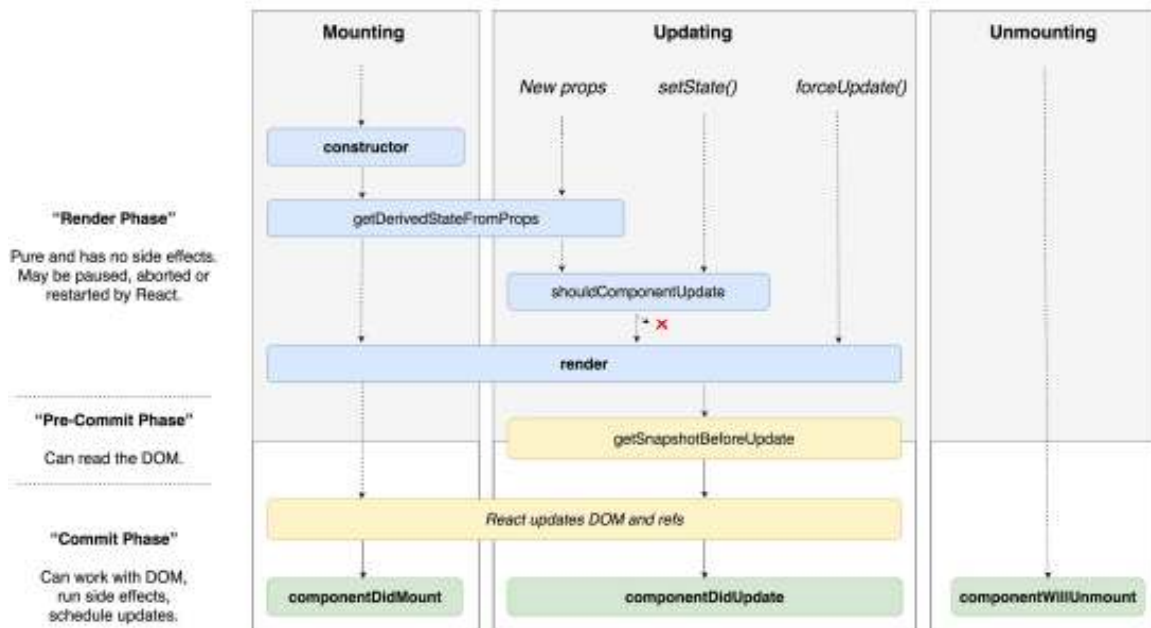
## > Lifecycle Methods

React

- Todos os componentes em React passam por um conjunto de etapas a que se dá o nome de métodos de ciclo de vida – **Lifecycle Methods**
  - Todos os componentes passam por três fases :
    - 1) *Mount* – Insere componentes no DOM
    - 2) *Update* – Envolva o métodos para atualizar os components no DOM
    - 3) *Unmount* – Remove o componente do DOM
- Permitem controlar e definir o workflow do componente, declarando e especificando código de acordo com os métodos disponíveis;
- Existe uma oordem específica pela qual estes métodos são executados
- Metodos muito utilizados em **Componentes de Class**:
  - *componentDidMount*
  - *componentDidUpdate*
  - *componentWillUnmount*
- Métodos muito utilizados em **Componentes Funcionais** com Hooks:
  - *useEffects*

## > Class Comp. > *Lifecycle Methods*

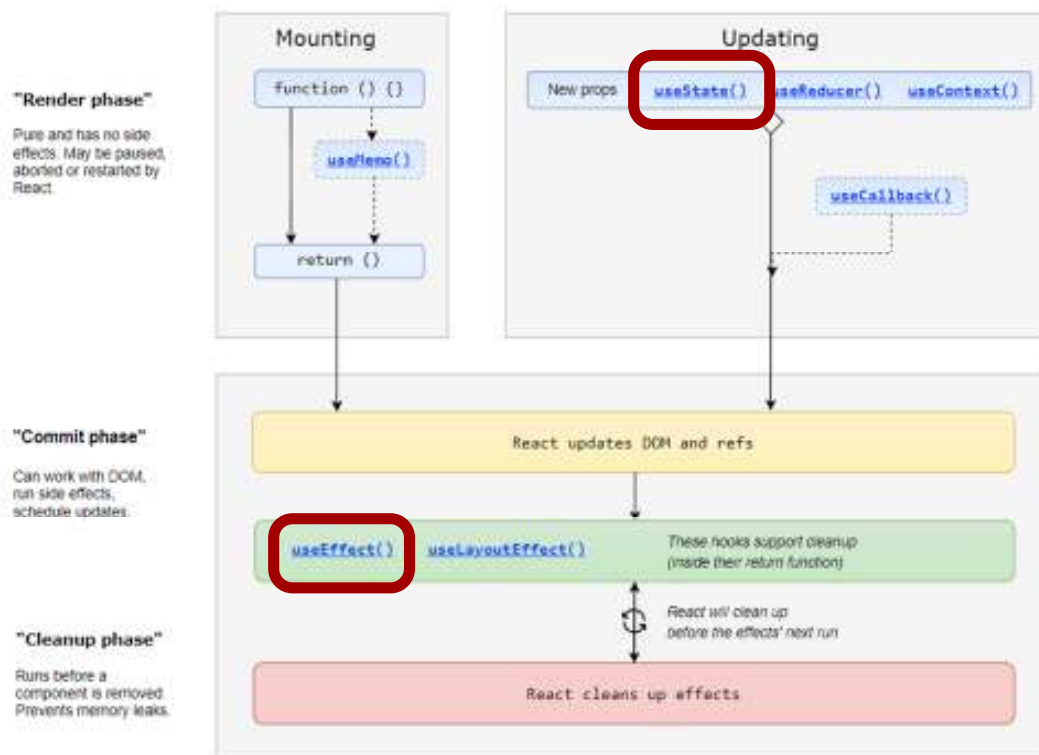
React



<https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/>

## > Func. e Hook > *Lifecycle*

React



<https://wavez.github.io/react-hooks-lifecycle/>

## > Func. e Hook > *Lifecycle*

React



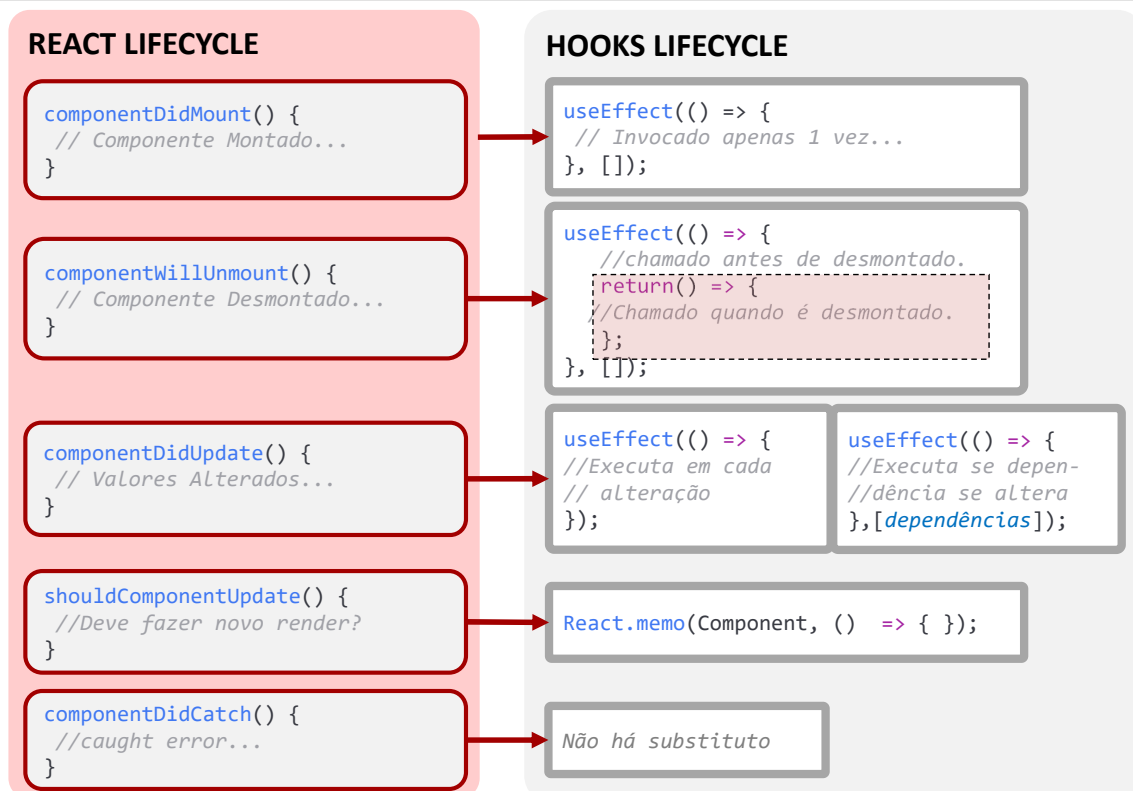
Notes:

1. Updates are caused by a parent re-render, state change, or context change.
2. Lazy initializers are functions passed to `useState` and `useReducer`.

<https://github.com/kentcdodds/react-hooks/blob/main/src/examples/hook-flow.png>

## > React vs Hooks Lifecycle

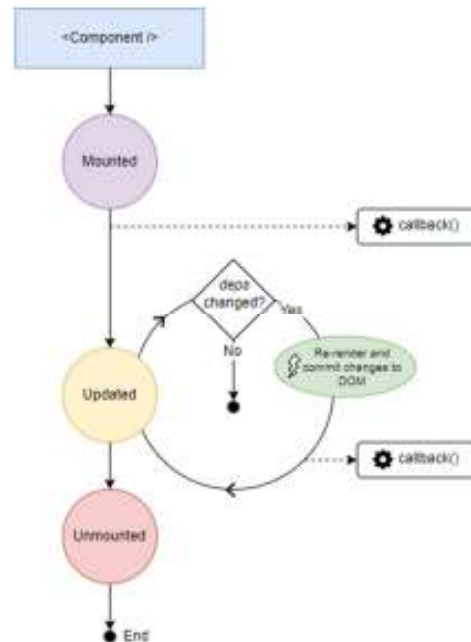
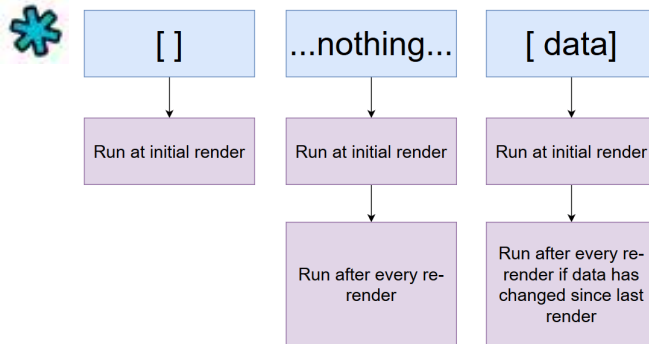
React



## > Hooks > useEffect

React

```
useEffect(function callback() {  
  return() => {  
    };  
}, *);
```



## > Comp. Funcionais > Hooks

React

- **useState**
  - Permite adicionar estados a componentes funcionais
- **useEffect**
  - Permite usar “lifecycle methods”
- **\*Outros:**
  - **useContext**
    - Permite usar contexto do sistema
  - **useRef**
    - Permite usar referencia
  - **useReducer**
    - Permite armazenar data através do “reducer”

## <Exemplo 1>



## > Class Component > Exemplo

React

```
import React from "react";
import "./styles.css";
class ContadorClass extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }
  componentDidMount() {
    this.setState({ count: this.state.count + 1 });
  }
  handleIncrementa = () => {
    this.setState({ count: this.state.count + 1 });
  };
  handleDecrementa = () => {
    this.setState({ count: this.state.count - 1 });
  };
  render() { return ( ... ); }
}
export default ContadorClass;
```

1

Incrementa Decrementa

Estado inicial

Coloca  
estado a 1

## > Class Component > Exemplo

React

```
...  
return (  
  <div className="counter">  
    <h1 className="count">{this.state.count}</h1>  
    <button type="button" onClick={this.handleIncrementa}>  
      Incrementa  
    </button>  
    <button type="button" onClick={this.handleDecrementa}>  
      Decrementa  
    </button>  
  </div>  
)  
...  

```

2

Incrementa Decrementa

## > Functional Component > Exemplo

React

```
import React, { useState, useEffect } from "react";  
import "./styles.css";  
  
const App = () => {  
  const [count, setCount] = useState(0);  
  useEffect(() => {  
    setCount((currentCount) => currentCount + 1)  
    , []);  
  
    const handleIncrementa = () =>  
      setCount((currentCount) => currentCount + 1);  
  
    const handleDecrementa = () =>  
      setCount((currentCount) => currentCount - 1);  
  
    return ( ... );  
  
  export default App;  

```

1

Incrementa Decrementa

Estado inicial

Coloca  
estado a 1

## > Functional Component > Exemplo

React

```
import React, { useState, useEffect } from "react";  
import "./styles.css";
```

1

```
const App = () => {
```

```
  const [count, setCount] =
```

```
    useEffect(() =>
```

```
      setCount((currentCount) => currentCount + 1),  
      []);
```

```
  const handleIncrementa
```

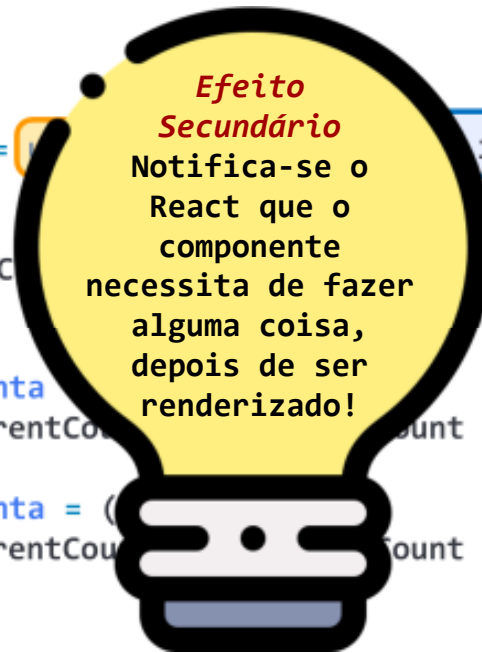
```
    setCount((currentCount) => currentCount + 1);
```

```
  const handleDecrementa =
```

```
    setCount((currentCount) => currentCount - 1);
```

```
  return ( ... );
```

```
  export default App;
```



## > Functional Component > Exemplo

...

```
return (
```

2

```
  <div className="counter">
```

```
    <h1 className="count">{count}</h1>
```

```
    <button type="button" onClick={handleIncrementa}>
```

```
      Incrementa
```

```
    </button>
```

```
    <button type="button" onClick={handleDecrementa}>
```

```
      Decrementa
```

```
    </button>
```

```
  </div>
```

```
);
```

...

## > Functional Component > Exemplo

React

```
import React, { useState, useEffect } from "react";  
import "./styles.css";
```

1

```
const App = () => {
```

```
  const [count, setCount] = useState(0);
```

```
  useEffect(() =>
```

```
    setCount((currentCount) => currentCount + 1)
```

```
  , []);
```

```
  const handleIncrementa = () =>
```

```
    setCount((currentCount) =>
```

```
  const handleDecrementa = () =>
```

```
    setCount((currentCount) =>
```

```
  return ( ... );
```

```
  export default App;
```

Garante que este hook apenas será disparado quando o componente monta e retorna a função, quando especificada, quando o componente é desmontado.

## > Functional Component > Exemplo

React

```
import React, { useState, useEffect } from "react";  
import "./styles.css";
```

1

```
const App = () => {
```

```
  const [count, setCount] = useState(0);
```

```
  useEffect(() =>
```

```
    setCount((currentCount) =>
```

```
  );
```

```
  const handleIncrementa = () =>
```

```
    setCount((currentCount) => currentCount + 1);
```

```
  const handleDecrementa = () =>
```

```
    setCount((currentCount) =>
```

```
  return ( ... );
```

```
  export default App;
```

Ciclo Infinito!  
Não pára de  
incrementar



## > Functional Component > Exemplo

React

```
import React, { useState, useEffect } from "react";
import "./styles.css";
```

```
const App = () => {
```

```
  const [count, setCount] = useState(0);
```

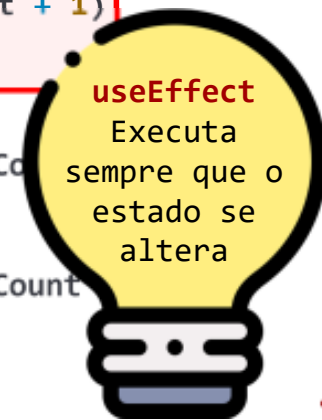
```
    useEffect(() => {
      setCount((currentCount) => currentCount + 1)
    });
```

```
    const handleIncrementa = () => {
      setCount((currentCount) => currentCount + 1);
```

```
    const handleDecrementa = () => {
      setCount((currentCount) => currentCount - 1);
```

```
    return ( ... );
```

```
  export default App;
```



isec  
Politécnico de Coimbra

## <Exemplo 2>



## > Functional Comp. > Exemplo 2

React

```
const App = () => {
  const [name, setName] = useState("Linguagens Script");
  const [visible, setVisible] = useState(true);

  return (<>
    {visible && <Ola name={name} />}
    <button onClick={() => setName("React")}>
      Mudar texto...
    </button>
    <button onClick={() => setVisible(false)}>
      Esconder....
    </button>
  </>);
};

export default App;
```

Ola, Linguagens Script!

Mudar texto...

Esconder....

isec  
Politécnico de Coimbra

## > Class Component > Exemplo 2

React

```
class Ola extends React.Component {
  componentDidMount() {
    console.log("<Ola> Mounted");
  } // é executado após o output do componente ser renderizado no DOM.

  componentWillUnmount() {
    console.log("<Ola> Removed");
  } // é usado para desmontagem de componentes (destruir)

  componentDidUpdate() {
    console.log("<Ola> props or state updated");
  } // é invocado imediatamente após a atualização ocorrer.
    // Não é chamado para a renderização inicial.

  render() {
    return <h1>Ola, {this.props.name}!</h1>
  }
}
```

isec  
Politécnico de Coimbra

## > Class Component > Exemplo 2

Ola, Linguagens Script!

Mudar texto... Esconder....

<Ola> Mounted

Ola, React!

Mudar texto... Esconder....

<Ola> props or state updated

Mudar texto... Esconder....

<Ola> Removed

Console was cleared

<Ola> Mounted

<Ola> props or state updated

<Ola> Removed

## > Functional Comp. > Exemplo 2

React

```
import React, { useState,useEffect } from "react";
const Ola = props => {
  useEffect(() => {
    console.log("<Ola> Mounted");
    return () => console.log("<Ola> is being removed");
  }, []);

  useEffect(() => {
    console.log("<Ola> props updated");
  }, [props]);
  return (<h1>Ola, {props.name}!</h1>);
};
```

<Ola> Mounted

<Ola> props updated

<Ola> props updated

<Ola> is being removed

---

## <Exemplo 3>

---



< 132 >

### > Functional Comp. > Exemplo 3

---

React

```
import "./styles.css";
import { useEffect, useState } from "react";
import { Ola } from "./Ola";
const NOMES = ["Jose", "Maria", "Filipa", "Nuno", "Manuel"];
export default function App() {
  const [index, setIndex] = useState(0);
  useEffect(() => {
    if (index === NOMES.length - 1) {
      return;
    }
    setTimeout(() => setIndex((index) => index + 1), 2000);
  }, [index]);
  return <Ola nome={NOMES[index]} />;
}
```



## > Functional Comp. > Exemplo 3

React

```
import { useEffect } from "react";
export function Ola({ nome }) {
  const mensagem = `Olá, ${nome}!`;
  console.log(`Render <Greet name="${nome}" />`);
  useEffect(() => {
    document.title = "Pagina Boas Vindas!";
    console.log("Executa Side-effect!");
  }, []);

  return <div>{mensagem}</div>;
}
```

React

# </Lifecycle Methods>



**</React>**

