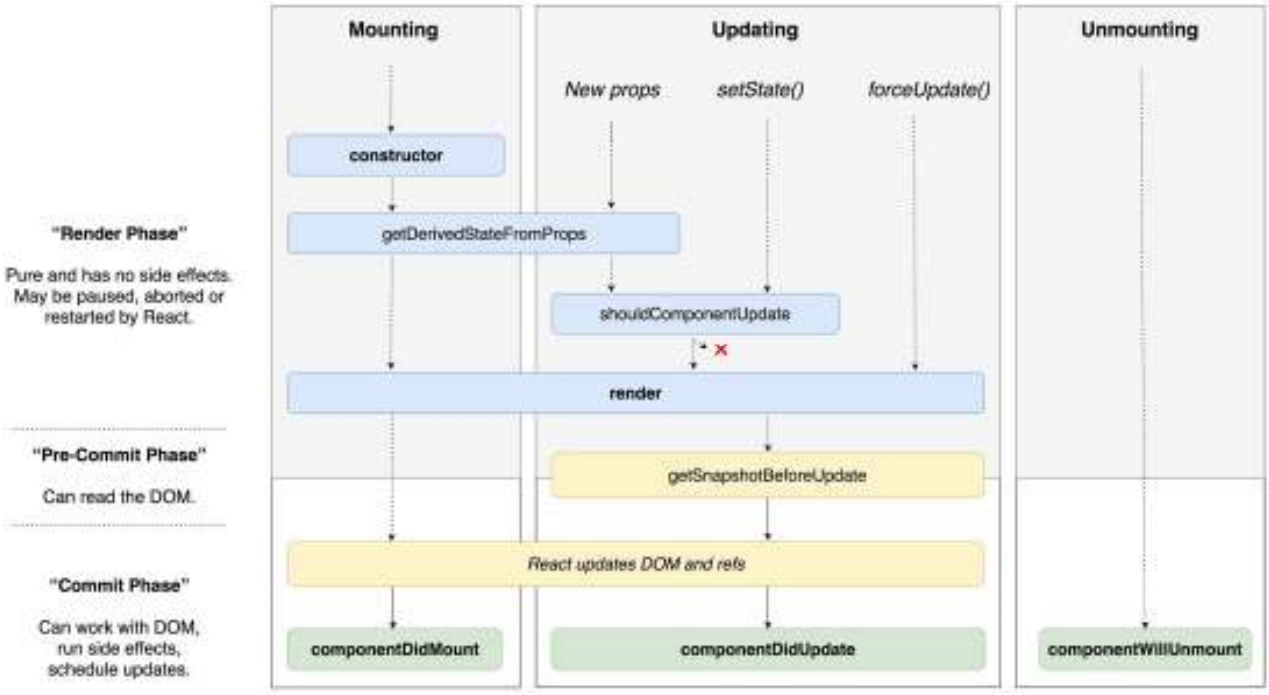
Lifecycle Methods

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

Lifecycle Methods

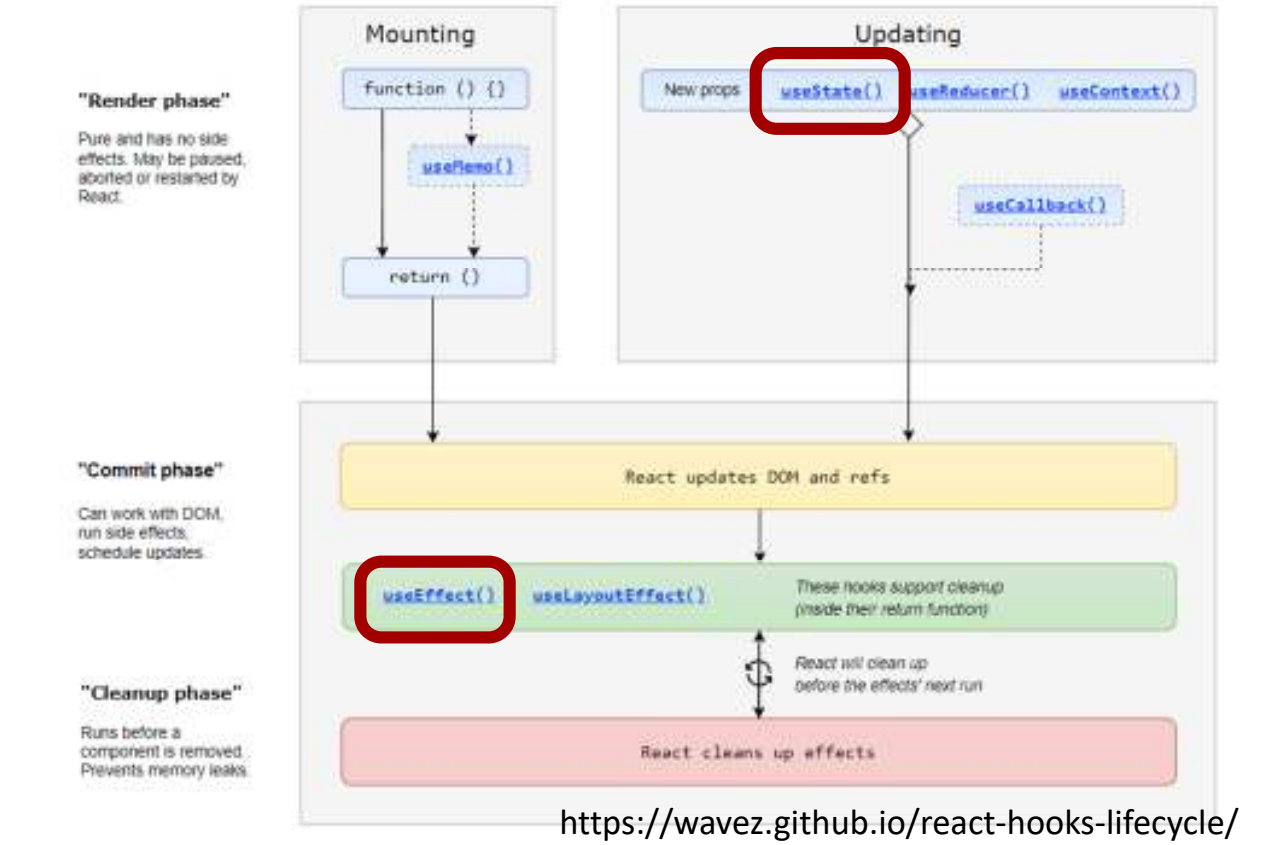
- Todos os componentes em React passam por um conjunto de etapas a que se dá o nome de métodos de ciclo de vida – **Lifecycle Methods**
  - Todos os componentes passam por três fases :
    - *Mount*
    - *Update*
    - *Unmount*
- Permitem controlar e definir o workflow do componente, declarando e especificando 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*



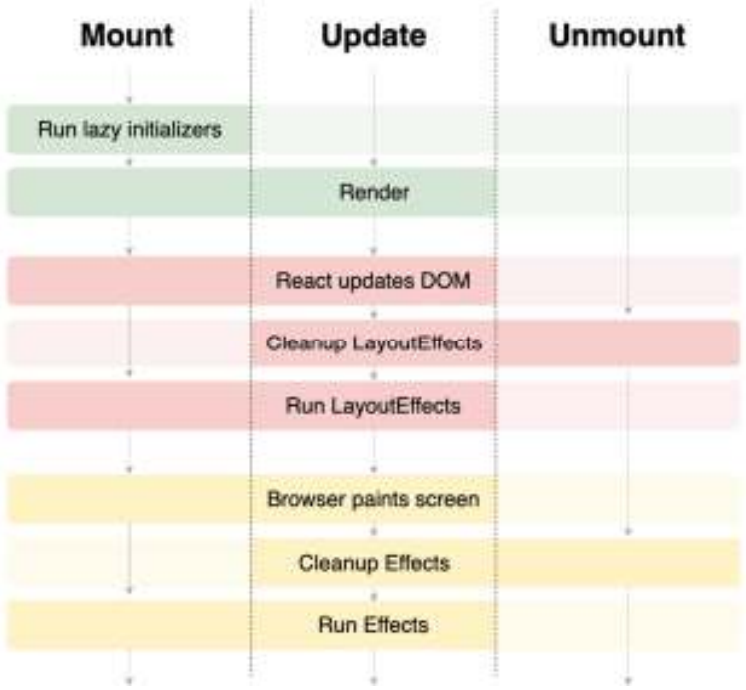
https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

Func. e Hook > *Lifecycle*



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

Func. e Hook > *Lifecycle*



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 LIFECYCLE

```
componentDidMount() {
 // Componente Montado...
}
```

```
componentWillUnmount() {
 // Componente Desmontado...
}
```

```
componentDidUpdate() {
 // Valores Alterados...
}
```

```
shouldComponentUpdate() {
 //Deve fazer novo render?
}
```

```
componentDidCatch() {
 //caught error...
}
```

HOOKS LIFECYCLE

```
useEffect(() => {
 // chamado apenas 1 vez...
}, []);
```

```
useEffect(() => {
   //chamado antes de desmontado.
   return() => {
   //Chamado quando é desmontado.
   };
}, []);
```

```
useEffect(() => {
//Executa em cada
// alteração
});
```

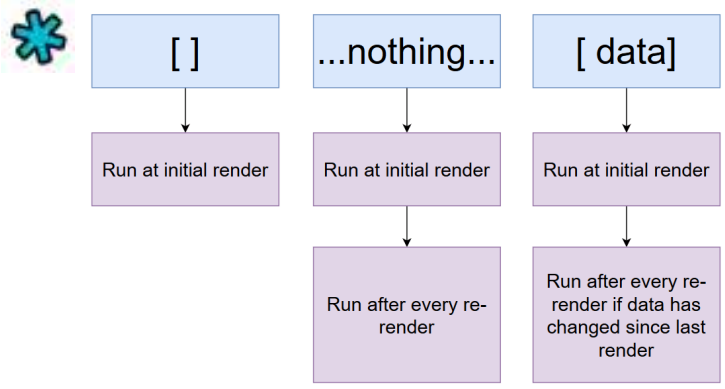
```
useEffect(() => {
//Executa se depen-
//dência se altera
},[dependências]);
```

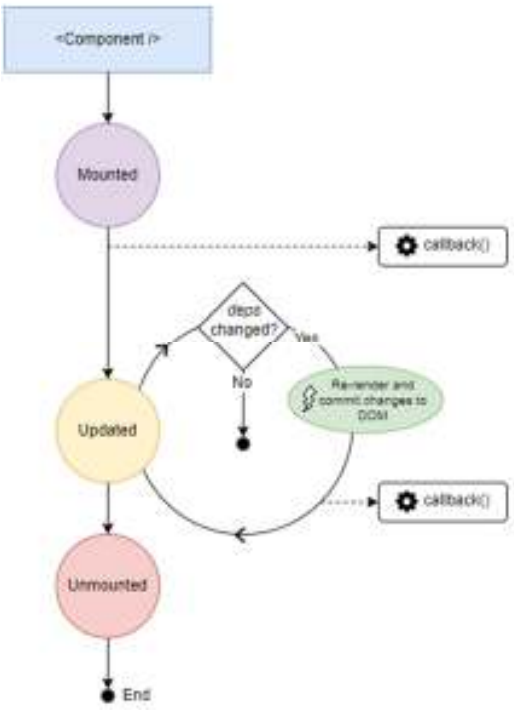
```
React.memo(Component, ()  => { });
```

Não há substituto

Hooks > useEffect


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





Comp. Funcionais > Hooks

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

<Exemplo 1>

> *Class Component* > Exemplo

```
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


```
...
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


```
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

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

const App = () => {

const [count, setCount] =

    useEffect(() =>
       setCount((currentC
    , []);

     const handleIncrementa
           setCount((currentCo

     const handleDecrementa = (
           setCount((currentCou

    return ( ... );

export default App;
```

1

Incrementa | Decrementa

inicial

Coloca estado a 1

Efeito Secundário
Notifica-se o React que o componente necessita de fazer alguma coisa, depois de ser renderizado!

unt + 1);

ount - 1);

Functional Component > Exemplo

```
...
return (
  <div className="counter">
     <h1 className="count">{count}</h1>
     <button type="button" onClick={handleIncrementa}>
              Incrementa
     </button>
     <button type="button" onClick={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 han
          set

    const han
          set

    return ( ... );

export default App;
```

1

Incrementa | Decrementa

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";

const App = () => {

const [count, setCount] = useSta

    useEffect(() =>
      setCount((currentCount) =>
    );

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

    const handleDecreme
          setCount((cur

    return ( ... );

export default App;
```

1

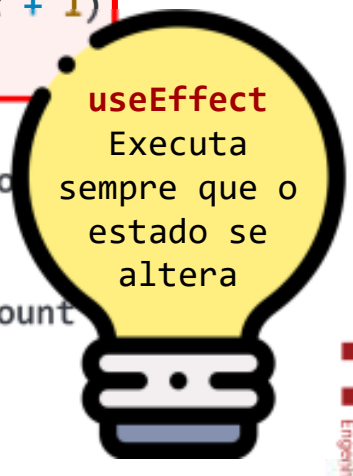
Incrementa | Decrementa

Ciclo Infinito!

Não pára de incrementar

Functional Component > Exemplo


```
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) => currentCo

    const handleDecrementa = () =>
          setCount((currentCount) => currentCount

   return ( ... );

export default App;
```



<Exemplo 2>

Functional Comp. > Exemplo 2


```
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;
```

Class Component > Exemplo 2


```
class Ola extends React.Component {
      componentDidMount() {
            console.log("<Ola> Mounted");
      } // é executado apóso 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>}
}
```

Class Component > Exemplo 2

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>);
};
```

<Exemplo 3>

Functional Comp. > Exemplo 3


```
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

```
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 Methos>