2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Application Delivery Fundamentals: React

Module 1: Introdução ao React

React Lifecycle

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

LifeCycle

React Lifecycle

LifeCycle

- O ciclo de vida do React é uma ferramenta importante para controlar e personalizar o comportamento dos componentes do **React**.

- Ao entender e usar os métodos de ciclo de vida, os desenvolvedores podem criar aplicativos **React** mais eficientes e com melhor desempenho

React Lifecycle

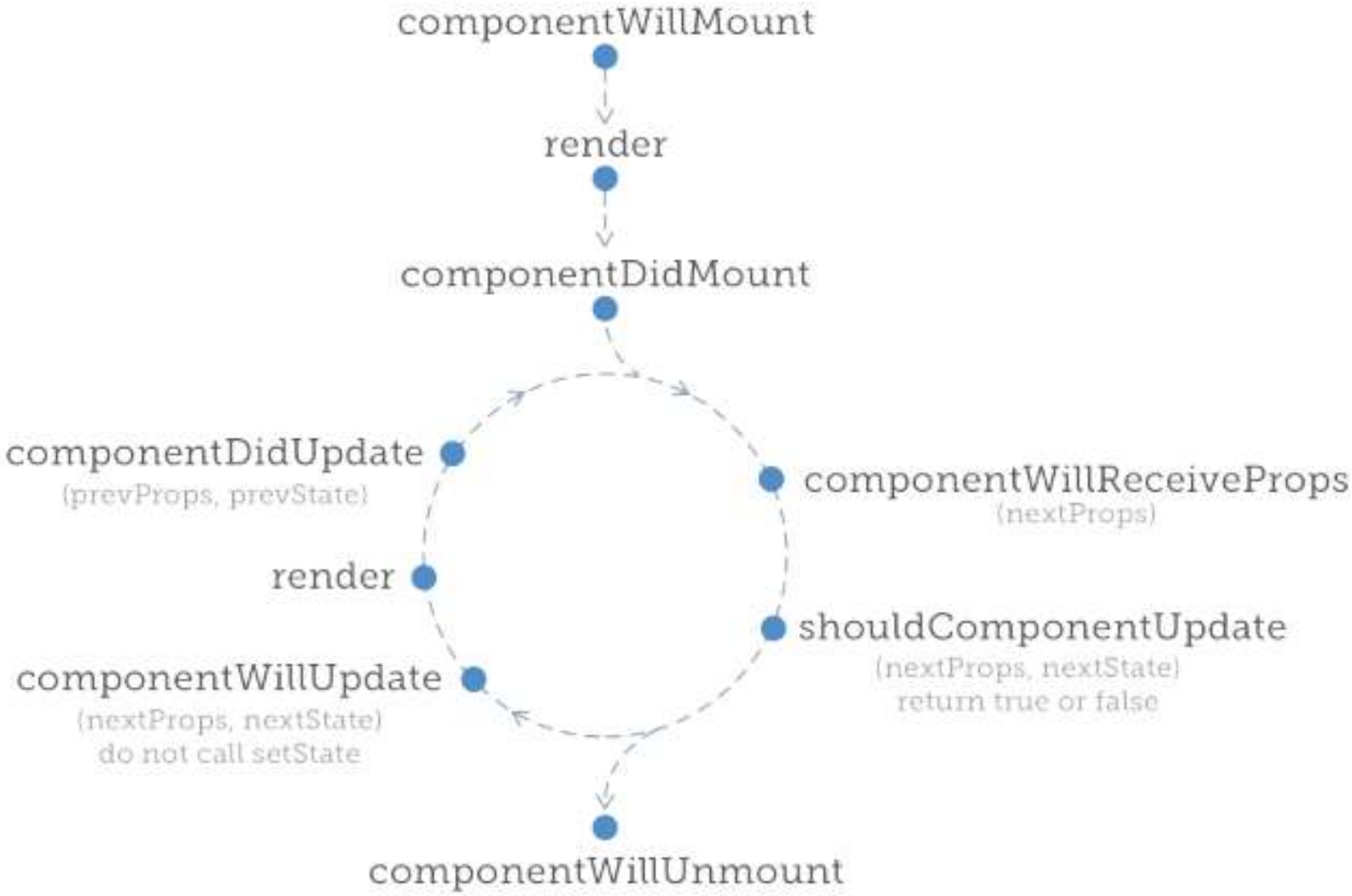
Benefícios do uso do ciclo de vida

- Otimização de desempenho: Isso pode ajudar a evitar novas renderizações desnecessárias e melhorar a velocidade geral e a capacidade de resposta do aplicativo.

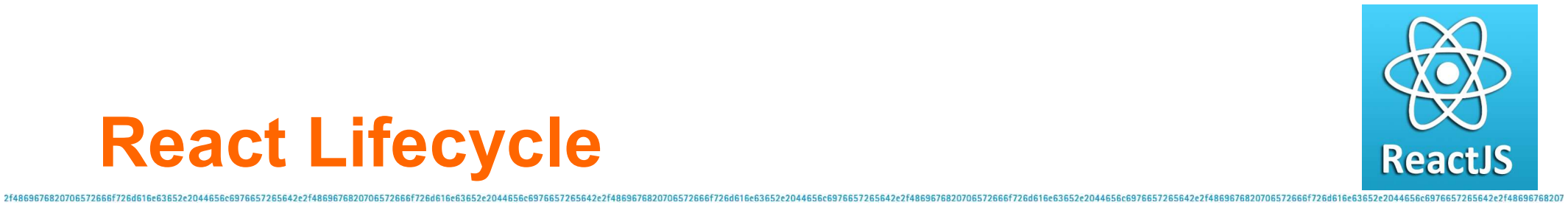
- Carregamento de dados: Isso permite que os componentes busquem e exibam dados com eficiência.

- Lógica de limpeza: Isso ajuda a evitar vazamentos de memória e garante que os componentes sejam limpos adequadamente.

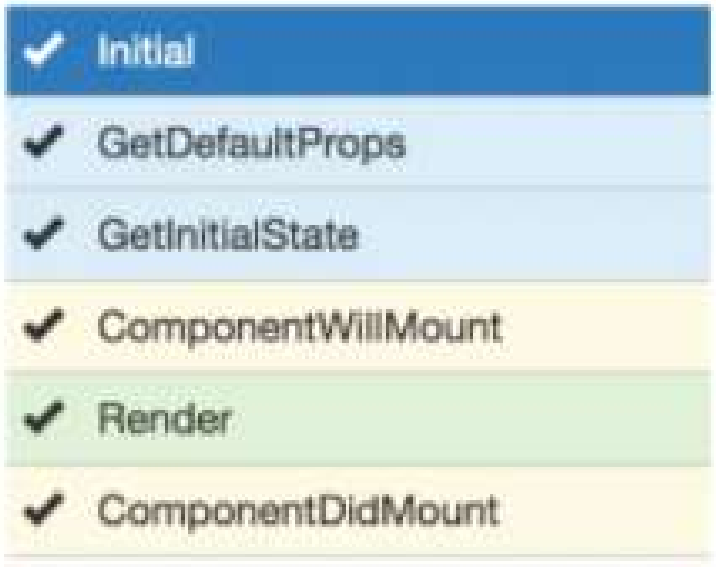
React Lifecycle



React Lifecycle

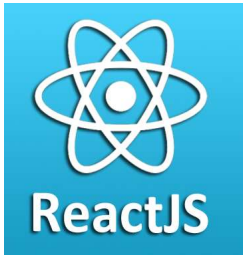


- Cada etapa do LifeCycle React possui uma função.



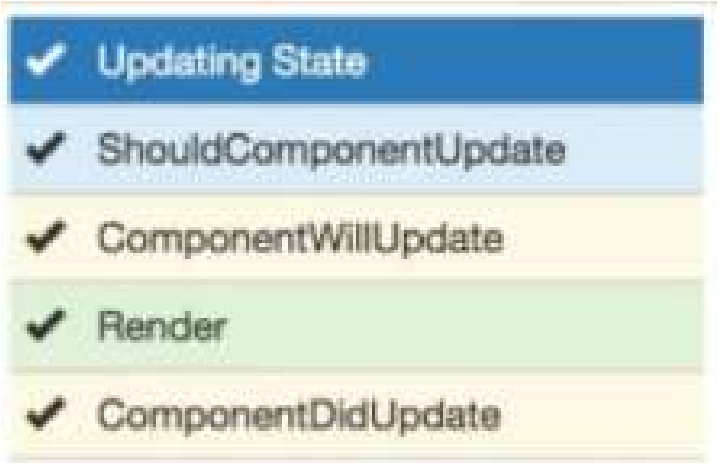
- Dentro do **ComponentWilMount** alteração no state não dispara re-render.
- Não podemos modificar um state no método render.
- Manipulação do DOM só é permitida dentro do **ComponentDidMount**.

React Lifecycle

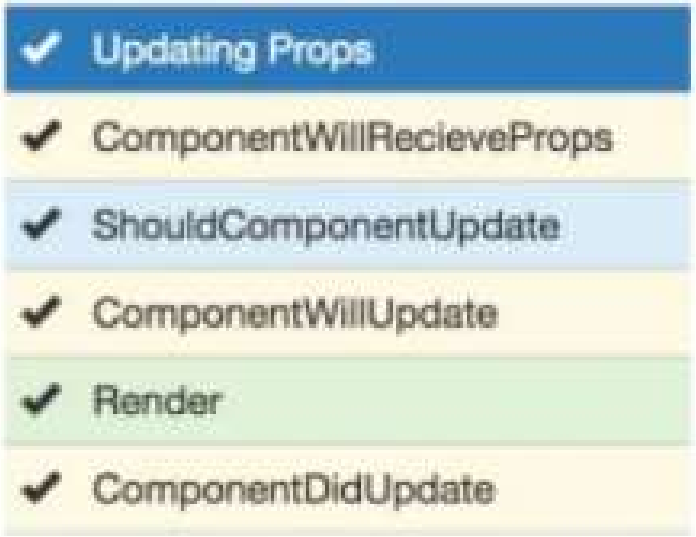


2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- Cada etapa do LifeCycle React possui uma função.



- Acontece quando o **state** é alterado.

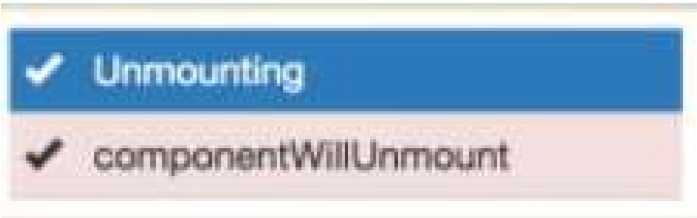


- Acontece quando o componente pai passa informações para o filho.
- **ComponentWillReceiveProps** não dispara re-render.

React Lifecycle

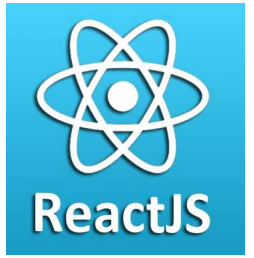
2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- Cada etapa do LifeCycle React possui uma função.



- Usado para apagar os dados.

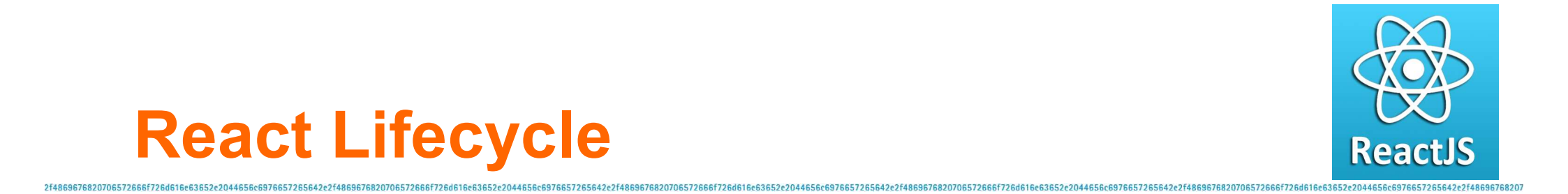
React Lifecycle



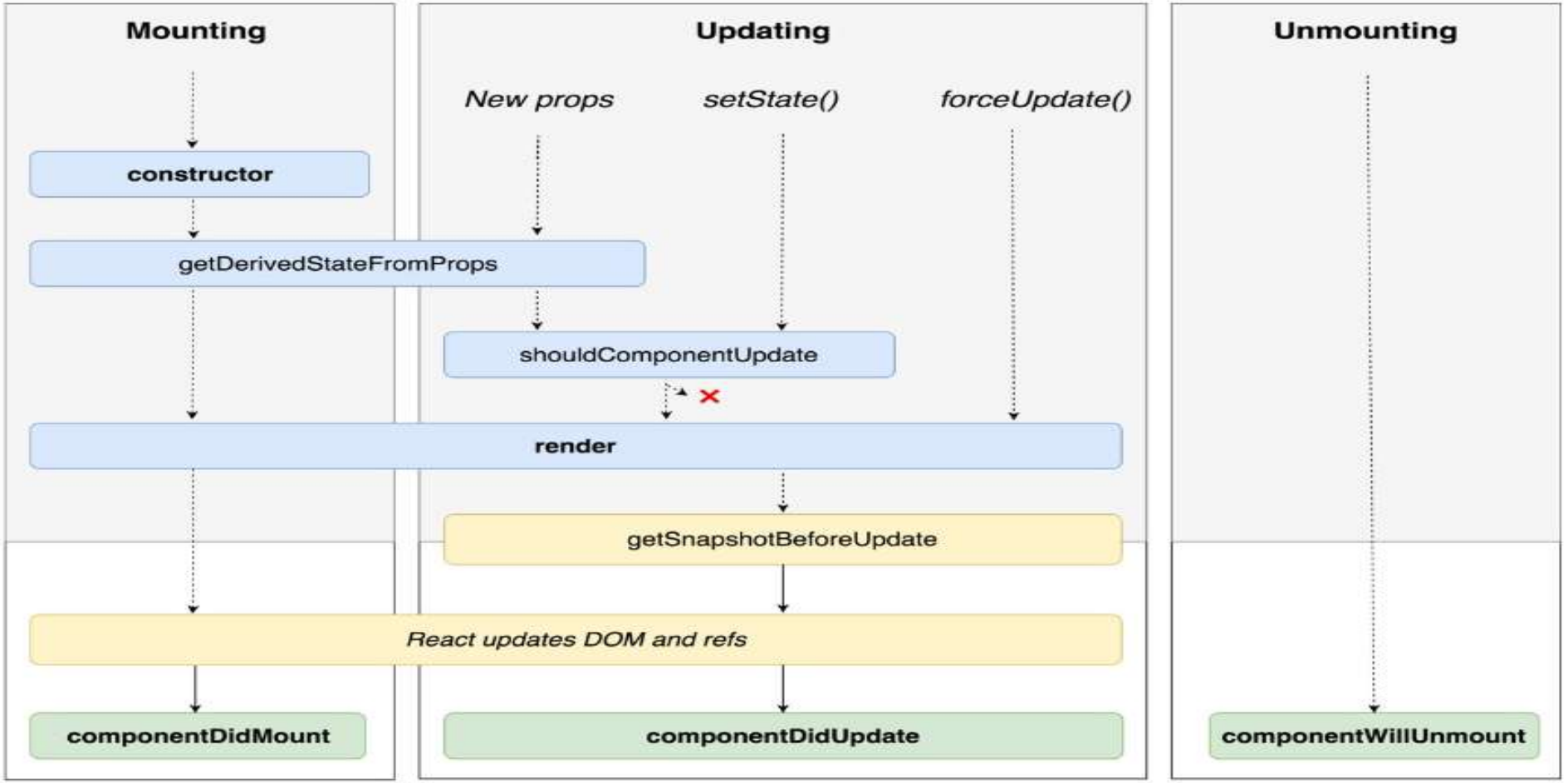
2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- Cada componente do React possui um ciclo de vida que você pode monitorar e manipular durante suas três fases principais.
  - As três fases são: **Mounting**, **Updating**, e **Unmounting**
    - **Mounting** significa colocar no **DOM;**
      - constructor() (LFCycle1.html)
      - getDerivedStateFromProps() (LFCycle2.html)
      - **render()**
      - componentDidMount() (LFCycle3.htm) e (LFCycle13.htm)
    - **Updating** Um componente é atualizado sempre que acontecer uma alteração no **state ou no props;**
      - getDerivedStateFromProps() (LFCycle4.html)
      - shouldComponentUpdate() (LFCycle5.html) (LFCycle6.html)
      - **render()** (LFCycle7.html)
      - getSnapshotBeforeUpdate() (LFCycle8.html)
      - componentDidUpdate() (LFCycle9.html)
    - **Unmounting** significa remover do **DOM;**
      - componentWillUnmount() (LFCycle10.html)

React Lifecycle

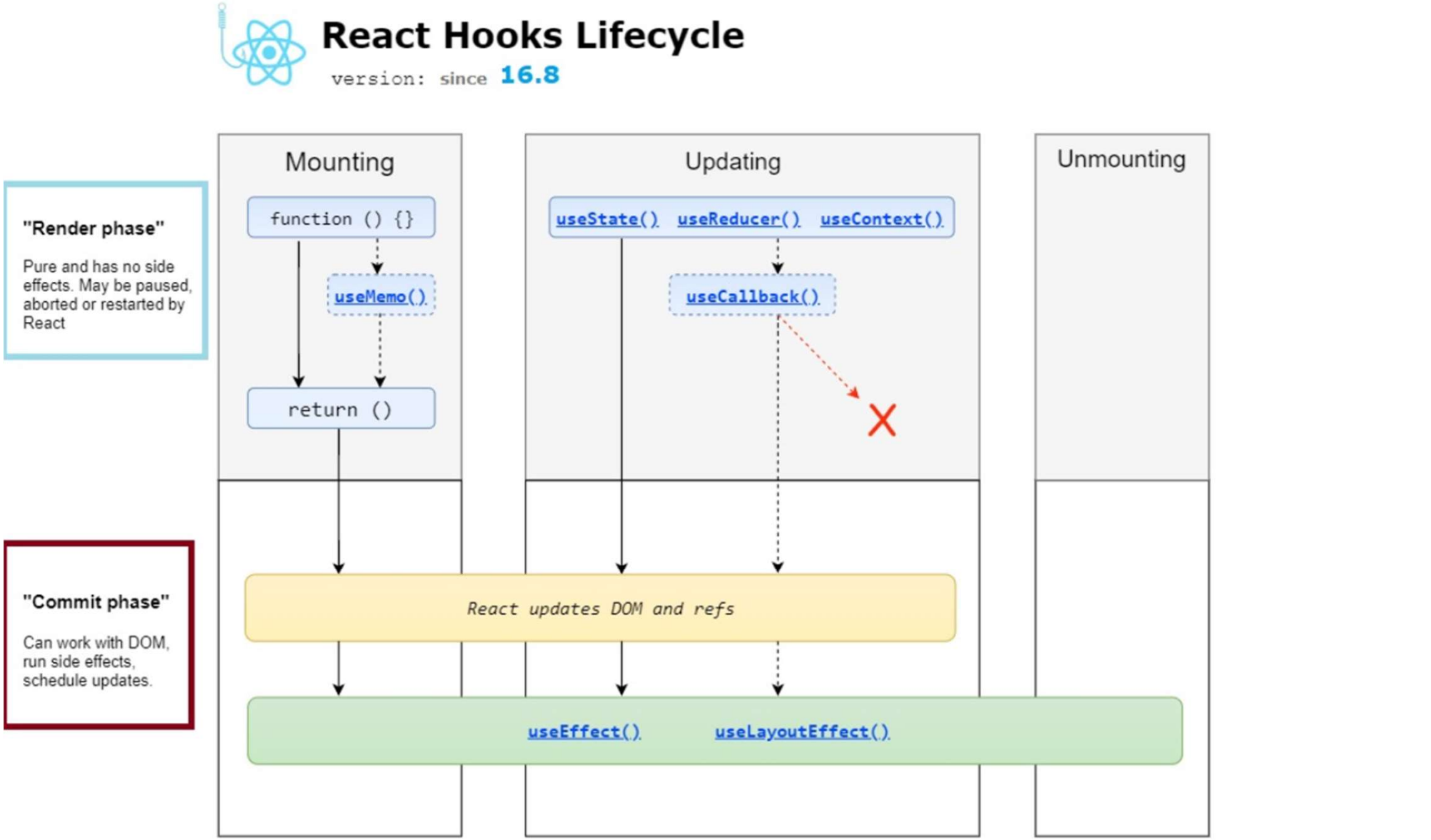


- https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/



- REACT_LFCycle.html, REACT_LFCycle11.html, REACT_LFCycle12.html

React Lifecycle



React Lifecycle

Mounting:
- useEffect(() => { ... }, []) // após o primeiro render

Updating:
- useEffect(() => { ... }, [dependencies]) // após alterações no state/props

Unmounting:
- useEffect Cleanup:

```
useEffect(() => {
  return () => {
    // código aqui
  };
}, [])
```

React Lifecycle

- Exercício:

- LFCycle12.html
- Altere o código acima renderizar o relógio sem parar;
  - componentDidMount()

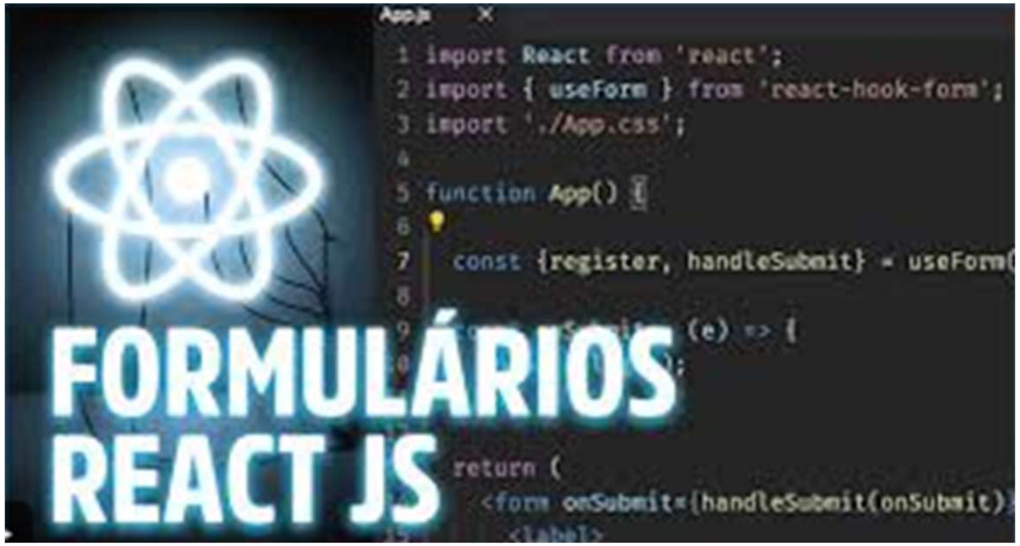
- LFCycle13.html
- Altere o código acima para permitir que ele pare o relógio.
  - componentDidMount()
  - componentWillUnmount()

- LFCycle14.html
- Altere o código acima para permitir que o relógio volte a funcionar no clique do mesmo botão;

React Form

2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207

- Assim como no HTML, o React usa formulários para permitir que os usuários interajam com a página da web;
- Em HTML, os dados do formulário geralmente são tratados pelo DOM;



- No React, os dados do formulário geralmente são tratados pelos componentes;

- Quando os dados são manipulados pelos componentes, todos os dados são armazenados no componente: **State**
  - (Form1.html) à (Form9.html)

React CSS

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- InLine Styling;
  - Para estilizar um elemento com o inline styling, o valor deve ser um objeto JavaScript;

- CSS Stylesheet
  - Você pode escrever seu estilo CSS em um arquivo separado, apenas salve o arquivo com a extensão .css e importe-o no seu aplicativo
  - (css1.html) à (css5.html)

React fragment

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

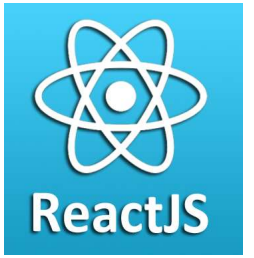
- Fragment;
  - Para exibir mais de um elemento no REACT ele necessita de um envelope para os elementos;
  - A Tag <DIV> é esse envelope;

```
return (
  <div>
    <h1 style={mystyle}>Hello Style!</h1>
    <p>Add a little style!</p>
  </div>
);
```

- <> </>
  - Você pode escrever um fragment para substuir a tag <DIV>
    - REACT_Fragment1.html
    - REACT_Fragment2.html
    - REACT_Fragment3.html

```
return (
  <>
    <h1 style={mystyle}>Hello Style!</h1>
    <p>Add a little style!</p>
  </>
);
```

React fragment



2f486967682070657266 6f726d616e63652e2044656c6976657265642e2f486967682070657266 6f726d616e63652e2044656c6976657265642e2f486967682070657266 6f726d616e63652e2044656c6976657265642e2f486967682070657266 6f726d616e63652e2044656c6976657265642e2f486967682070657266 6f726d616e63652e2044656c6976657265642e2f486967682070657207

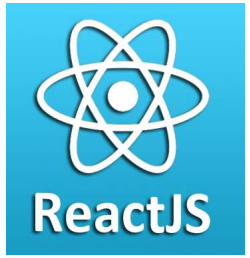
Class component

```
import React, { Component } from 'react';
export default class App extends Component {
    constructor(props) {
        super(props);
        this.state = {
            todos: [
                { id: 1, item: 'Wash cloth' },
                { id: 2, item: 'Fix bugs' },
            ],
        };
    }

    render() {
        return (
            <div>
                {this.state.todos.map((todo) => {
                    return <p key={todo.id}>{todo.item}</p>;
                })}
            </div>
        );
    }
}
```

Functional component

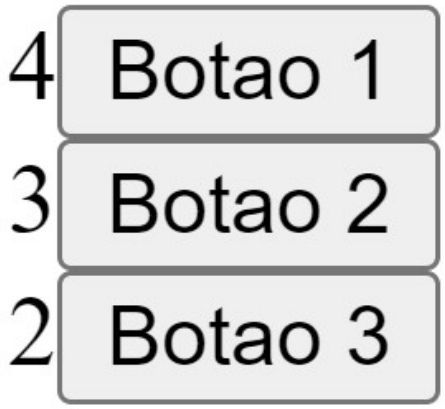
```
import React, { useState } from 'react';

const App = () => {
    const [todos, setTodos] = useState([
        { id: 1, item: 'Wash cloth' },
        { id: 2, item: 'Fix bugs' },
    ]);

    return (
<div>
    {todos.map((todo) => {
        return <p key={todo.id}>{todo.item}</p>;
    })}
</div>
    );
};

export default App;
```

Atividade

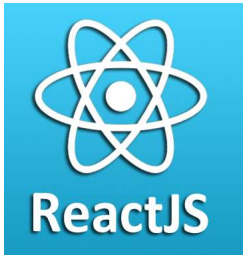


2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- Atividade (2 códigos)
- 1º. Código
  - Altere o REACT_Fragment3.html
  - Os três botões devem atualizar o mesmo contador
- 2º. Código
  - Refatore com functions



React Hooks



2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f486967682070

- Hooks;
  - *Hooks* permitem que você use o **state** e outros recursos do React sem escrever uma classe.;

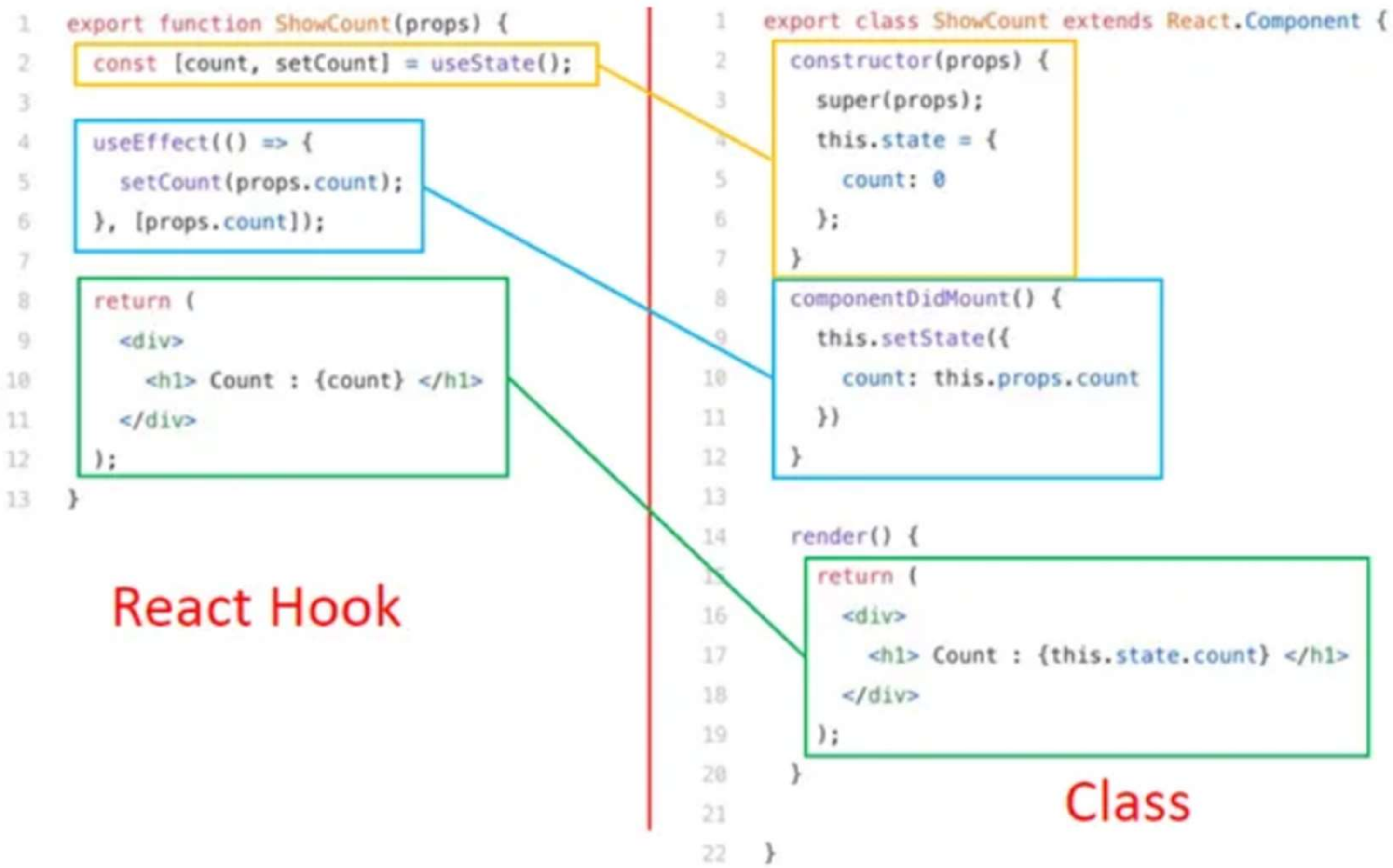
```
<script type="text/babel">

    const { useState } = React;

    function App() {
      const [count, setCount] = useState(0);
      return(
        <div>
            <p> Voce clicou {count} vezes </p>
            <button onClick={() => setCount(count+1)}>
               Me clica
            </button>
        </div>
      );
    }

    ReactDOM.render(<App />, document.getElementById('root'));
</script>
```

  - REACT_hooks1.html

React Hooks

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

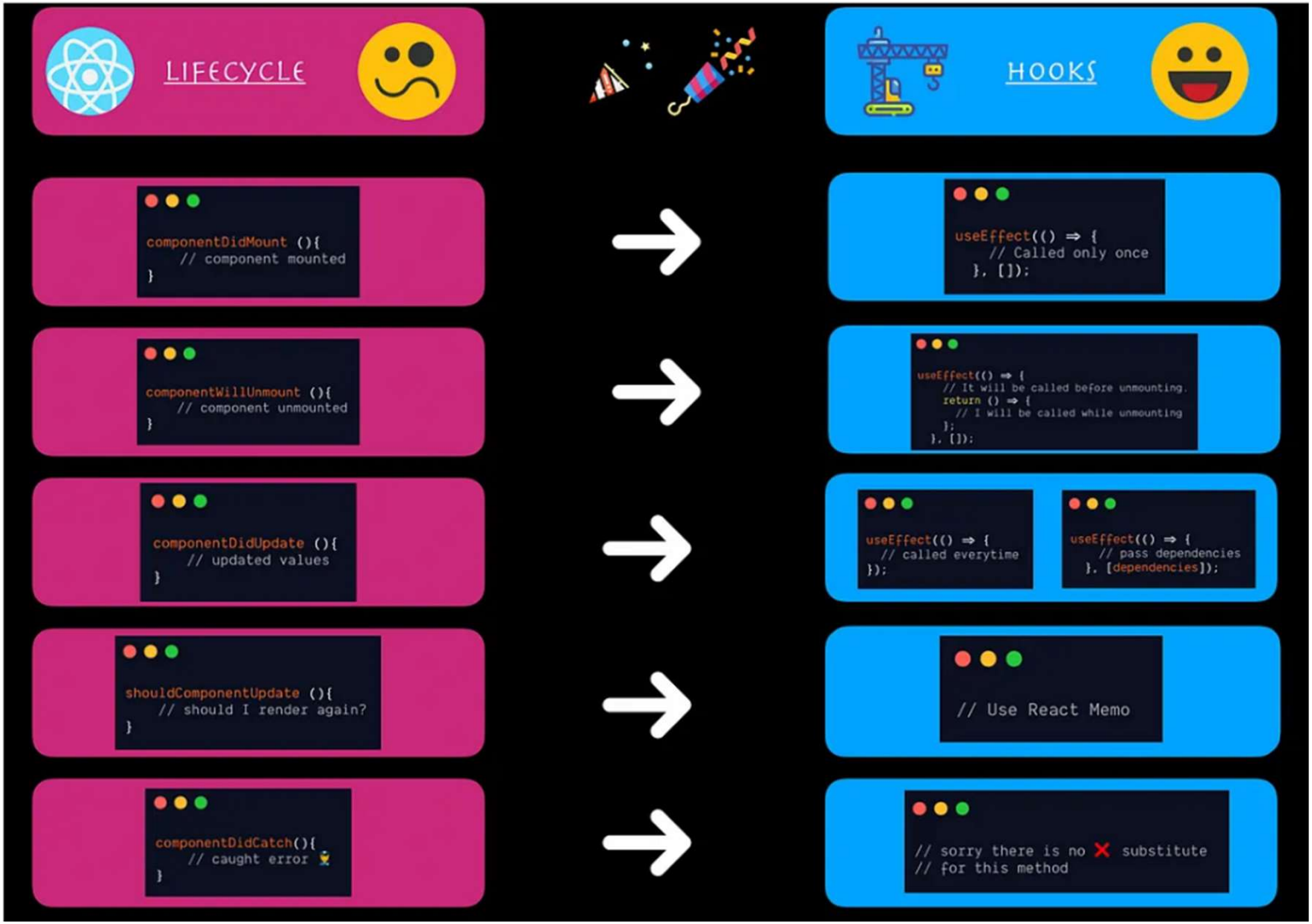
- Hooks;
  - *Life Cycle*.



React Hooks

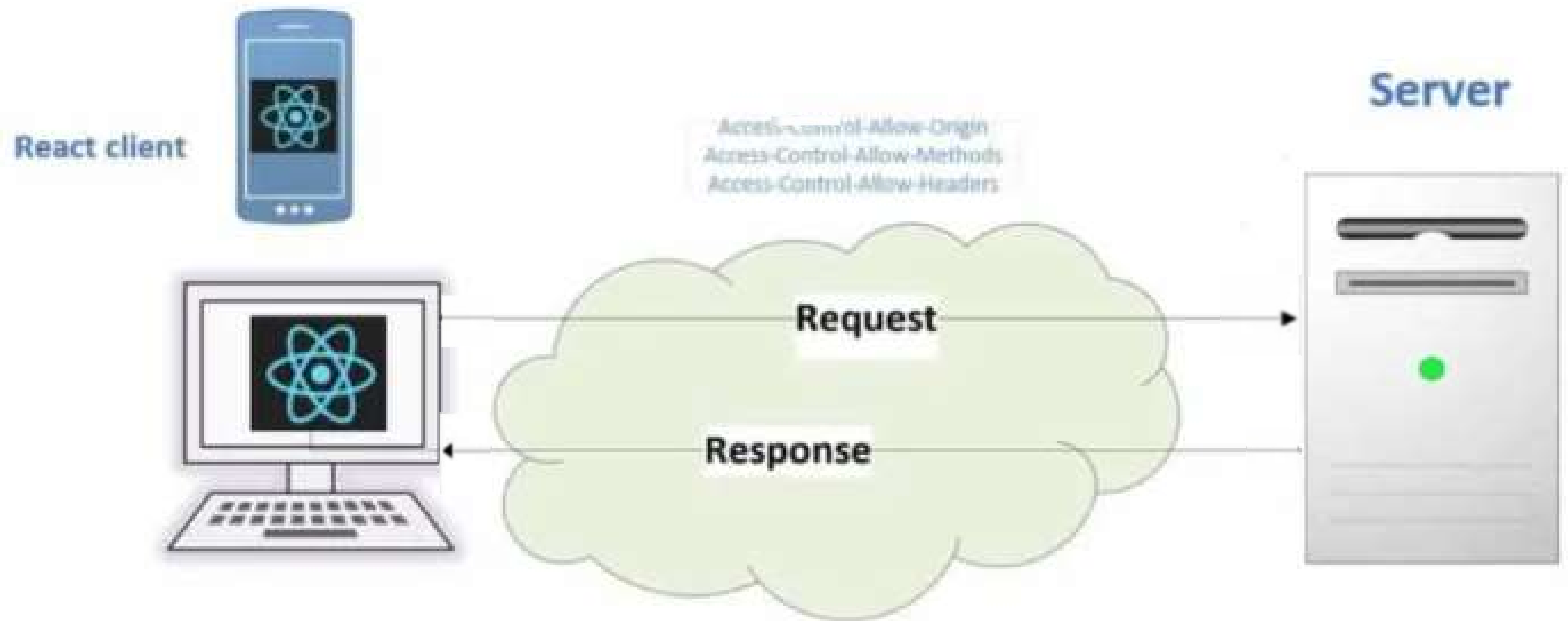
2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820

- Hooks;
  - *Life Cycle.*



React Fetch

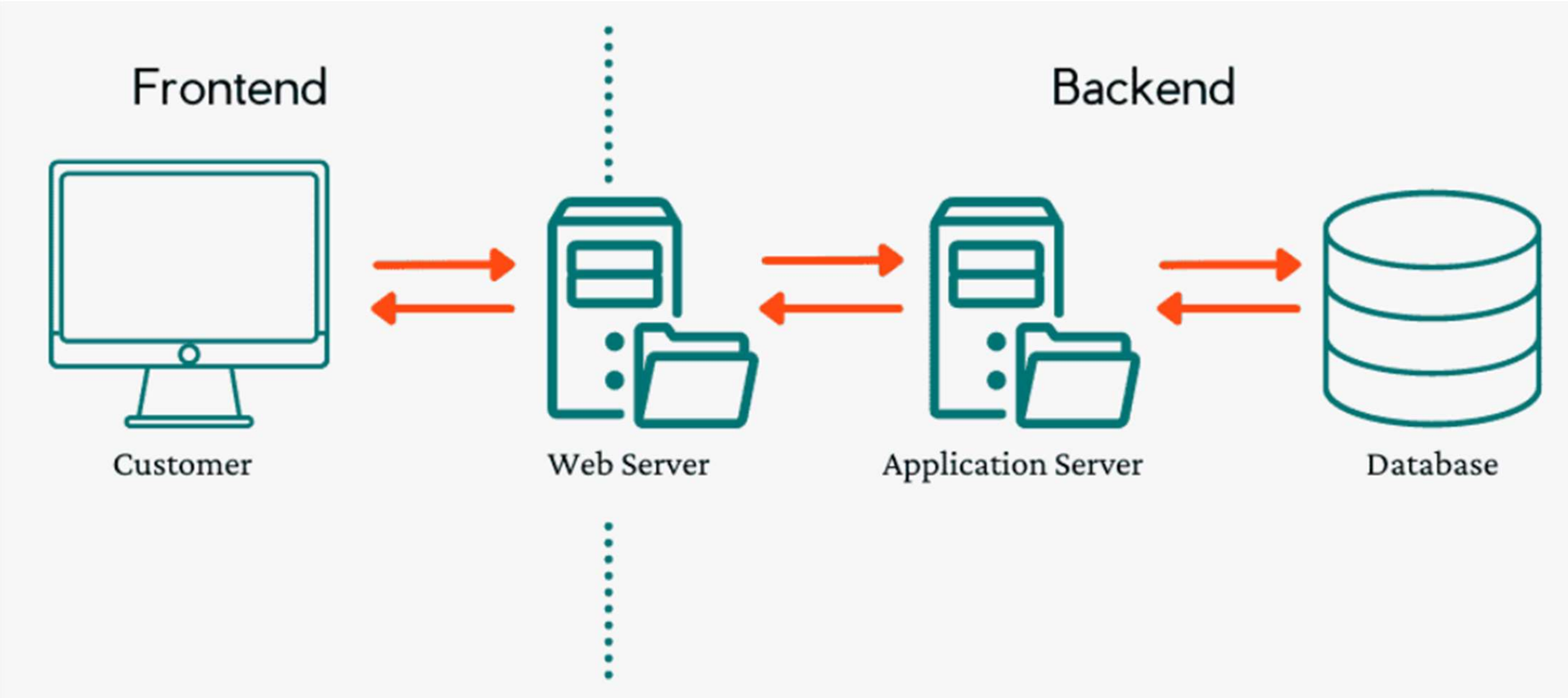
- Falando com o BACK-END;
  - *Request.*
  - *Response.*



React Fetch

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

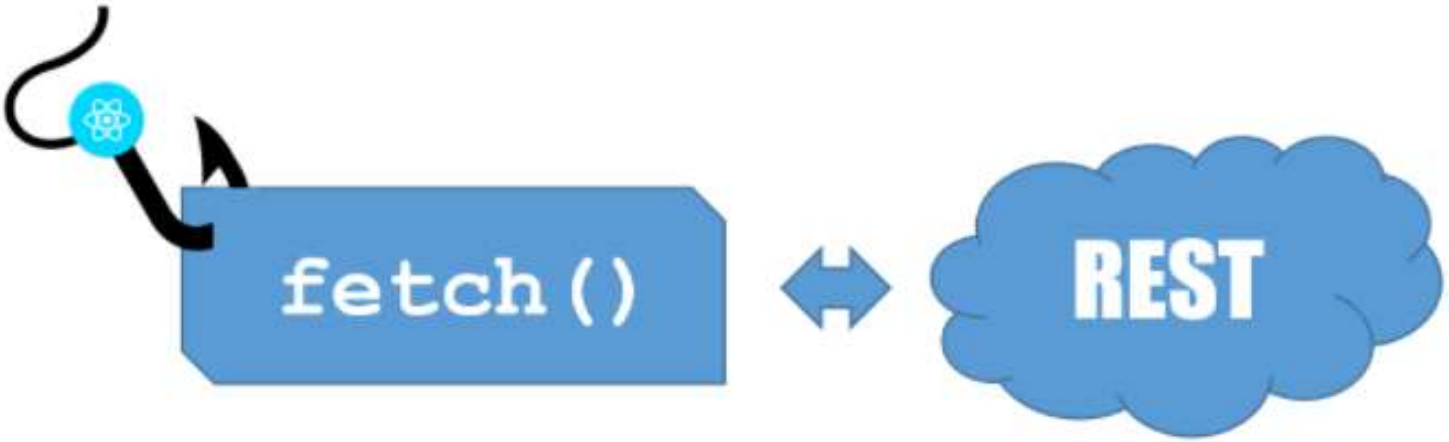
- Falando com o BACK-END;
  - *Request.*
  - *Response.*



React Fetch

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- Fetch;
  - Busca recursos de forma assíncrona através da rede.
  - *Response.*



React Fetch

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- Sintaxe do Fetch;

```
fetch('https://localhost:3000/alunos')
.then(response => response.json())
.then(data => console.log(data));
```

- No código, buscamos dados de uma URL que retorna dados JSON e, em seguida, imprimindo-os no console.

React Fetch

- Sintaxe do Fetch;

```
fetch(endPoint, requestOptions)
.then(function(response) {
        if (response.ok) {
                console.log('Resposta da Nuvem foi ok.');
          } else {
                console.log('Deu ruim na resposta da Nuvem.');
          }
        })
.catch(function(error) {
        console.log('Aconteceu um erro na operação de fetch: ' + error.message);
});
```

React Fetch

- Sintaxe do Fetch;

requestOptions

```
method: 'POST', // *GET, POST, PUT, DELETE, etc.
mode: 'cors', // no-cors, *cors, same-origin
headers: {
        'Content-Type': 'application/json'
       },
body: JSON.stringify({ id: '123', name : 'qweq’ })
```

React Fetch

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- Passos necessário para usar serviços HTTP no react:

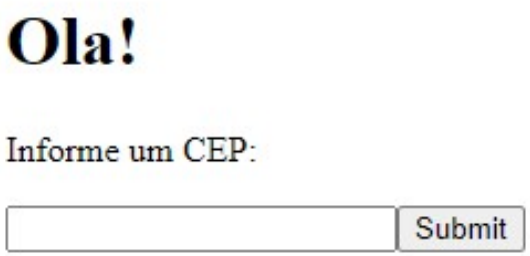
1. Crie um React Component, baseado em function ou class.
2. Monte o Fetch.
3. Execute.

    – REACT_Request1.html, 2 e 3

Atividade

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- Atividade:
  - Recupere e imprima o Logradouro do cep: 01001000



URL: viacep.com.br/ws/01001000/json/


```
{
  "cep": "01001-000",
  "logradouro": "Praça da Sé",
  "complemento": "lado ímpar",
  "bairro": "Sé",
  "localidade": "São Paulo",
  "uf": "SP",
  "ibge": "3550308",
  "gia": "1004",
  "ddd": "11",
  "siafi": "7107"
}
```