React Eventos e Hooks

Eventos

Os eventos em React são bem parecidos com a forma em que inserimos eventos no Javascript Vanilla dentro do HTML, com a diferença que utilizamos o **camelCase** ao chamar o evento e a função é chamada **dentro das chaves e sem parenteses**.

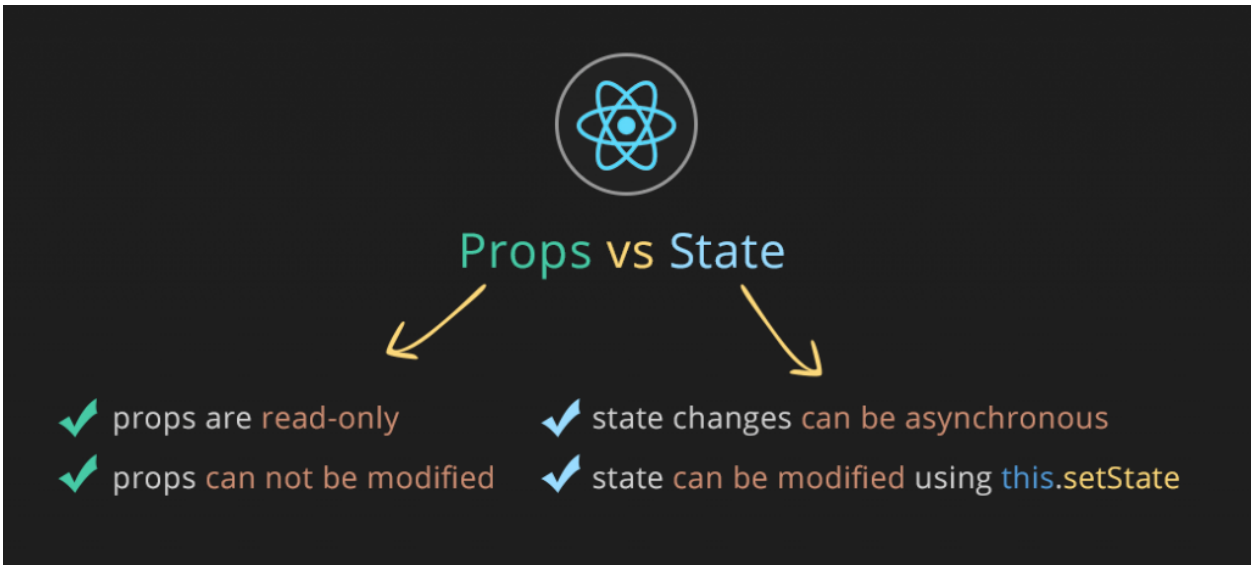
```
<button onClick={handleClick}>
    Clique aqui!
</button>
```

React Hooks

São funções especiais que deixam você “enganchar” features do React dentro de componentes funcionais.

- useState
- useEffect

UseState



O estado de um componente diz respeito à um componente que varia com o decorrer do tempo e interações de um usuário.

Toda vez que uma mudança é feita no estado ou props de um componente ele é renderizado novamente.

```
const [state, setState] = useState(initialValue)
const [count, setCount] = useState(initialCount)
const [anything, setAnything] = useState(initialAnything)
```

O exemplo mais utilizado em tutoriais é do contador, que permite aumentar um número ao clicar no botão:

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

    const aumentarValor = () => {
        setCount(count + 1);
    };

    return (
```

```
        <div>
            <p>{count}</p>
            <button type="button" onClick={aumentarValor}>
                Somar 1
            </button>
        </div>
    );
```

Você pode criar diversos estados, para todos as variáveis que você desejar:

```
const App = () => {
    const [name, setName] = useState("John Doe");
    const [age, setAge] = useState(20);
    const [hobby, setHobby] = useState("reading");

    return (
        // ...
    );
};
```

Você também poderia combinar os estados dentro de um objeto e acessá-los posteriormente:

```
const App = () => {
    const [userDetails, setUserDetails] = useState({
        name: 'John Doe',
        age: 20,
        hobby: 'Reading',
    });

    return (
        <div>
            <h1>{userDetails.name}</h1>
            <p>
```

```
                {userDetails.age} || {userDetails.hobby}
            </p>
        </div>
    );
```

Você também pode iniciar um estado como uma função, por exemplo quando você precisa fazer alguns cálculos para determinar como ele deve iniciar:

```
const expensiveComputation = () => {
    let number = 50;
    let newNumber = (50 % 10) * 10 - number;
    return newNumber;
};

const App = () => {
    const [calc, setCalc] = useState(() => expensiveComputation
    return (
        <div>
            <p>{calc}</p>
        </div>
    );
```

Regra para utilizar o useState

Você só pode chamar o useState() diretamente dentro do componente. Você não pode utilizá-lo dentro de uma função, loop, funções dentro do seu componente ou condicionais.

```
// Nunca faça isso ❌❌❌❌

if(condition){
    const [count, setCount] = useState()(0);
}

for (let index = 0; index < 25; index++) {
```

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

const nestedFn = () => () => {
    const [count, setCount] = useState()(0);
}
```

useEffect

O useEffect te permite lidar com efeitos colaterais no seu componente. Exemplo: pegar dados de API, alterar o DOM diretamente e timers.

Use setTimeout() para depois de 1 segundo que o componente for renderizado, alterar o contador para 1:

```
import { useState, useEffect } from "react";
import ReactDOM from "react-dom/client";

function Timer() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    setTimeout(() => {
      setCount((count) => count + 1);
    }, 1000);
  });

  return <h1>I've rendered {count} times!</h1>;
}
```

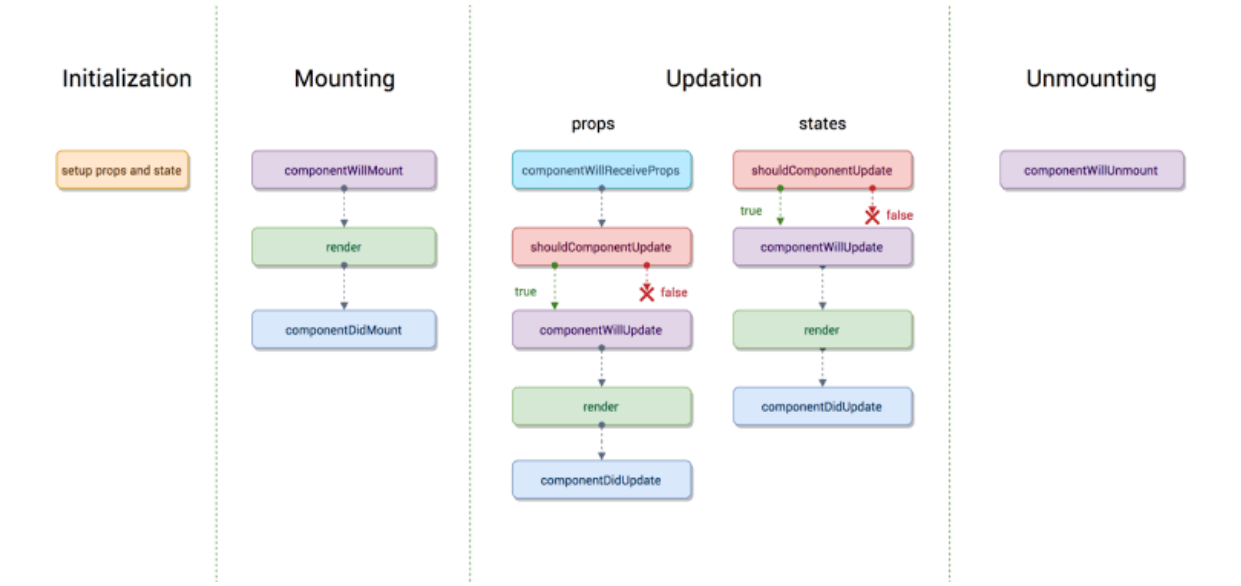
Utilizando na chamada de API:

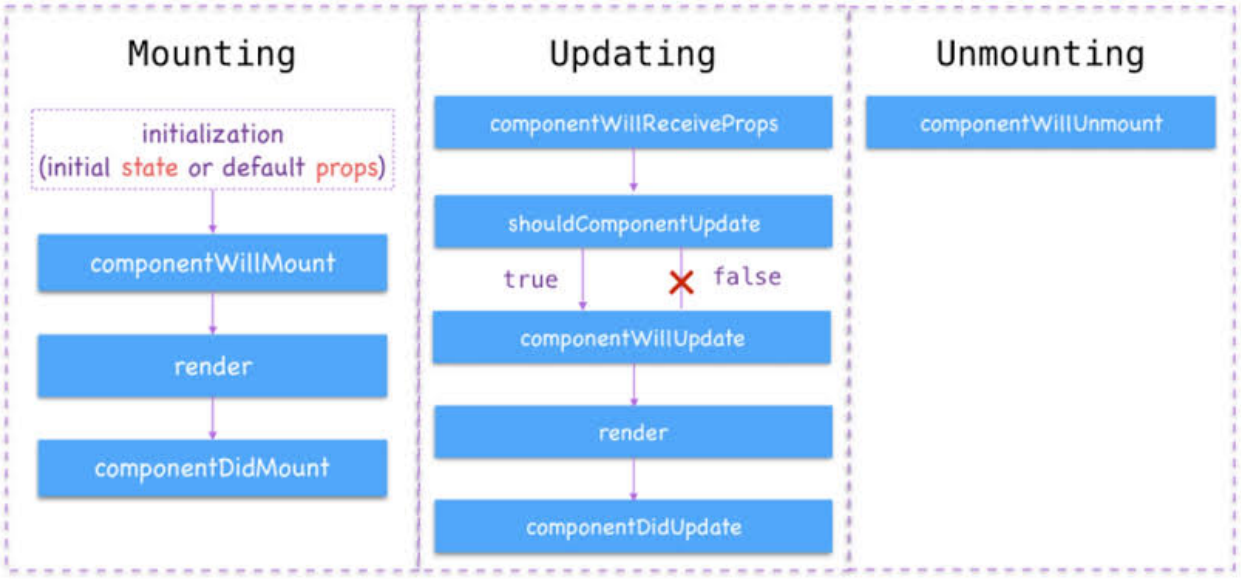
```
useEffect(()=> {
fetch('https://jsonplaceholder.typicode.com/photos')
.then(response => response.json())
```

```
  .then(json => console.log(json))
},[])
```

Ciclo de vida React

Os componentes em React possuem um ciclo de vida que nós podemos manipular durante 3 principais fases: **Mounting, Update e Unmounting.**





Mounting: quando a página carrega, o elemento é criado e inserido no DOM.

updating: quando o estado atualiza

unmounting: quando o usuário deixa a página

Exemplo

https://codepen.io/adamaoc/pen/ZQvbyX?editors=1111

Lidando com formulários

```
import { useState } from 'react';
import ReactDOM from 'react-dom/client';

function MyForm() {
  const [name, setName] = useState("");

  const handleSubmit = (event) => {
    event.preventDefault();
    alert(`The name you entered was: ${name}`)
  }

  return (
    <form onSubmit={handleSubmit}>
      <label>Enter your name:
        <input
          type="text"
          value={name}
          onChange={(e) => setName(e.target.value)}
        />
      </label>
      <input type="submit" />
    </form>
  )
}
```

```
const root = ReactDOM.createRoot(document.getElementById('root'
root.render(<MyForm />);
```

Referências

https://www.alura.com.br/artigos/react-hooks

https://dev.to/pratham10/all-you-need-to-know-about-react-hooks-54p0

https://youtu.be/jK0uiQ1ZQQQ?si=i2PPSyHhzMU9dgLU

https://www.geeksforgeeks.org/react-js-events/

https://blog.logrocket.com/react-lifecycle-methods-tutorial-examples/

https://www.w3schools.com/react/react_useeffect.asp

https://hygraph.com/blog/usestate-react

https://www.freecodecamp.org/news/react-component-lifecycle-methods/

https://engcfraposo.medium.com/entendendo-os-side-effects-e-o-ciclo-de-vida-do-react-5cc7c32bc43c

https://www.freecodecamp.org/news/how-to-fetch-api-data-in-react/

https://www.geeksforgeeks.org/react-js-events/

https://blog.logrocket.com/react-onclick-event-handlers-guide/

https://www.freecodecamp.org/portuguese/news/como-fazer-o-fetch-dos-dados-em-react/