Aula Componentes funcionais e componentes de classe

Os componentes podem retornar uma função ou componentes de classe.

Em versões antigas e componentes que tem estado devem ser classes e componentes que não tem estados podem ser funções ou classes sem estado também.

Não precisamos mais saber a diferença entre stateless e statefull.

Hoje se utiliza os hooks para controlar o estado.

Exemplo de componente função:

import logo from './logo.svg';

import './App.css';

function App() {

  return (

    <div className="App">

      <header className="App-header">

        <img src={logo} className="App-logo" alt="logo" />

        <p>

          Edit <code>src/App.js</code> and save to reload.

        </p>

        <a

          className="App-link"

          href="https://reactjs.org"

          target="\_blank"

          rel="noopener noreferrer"

        >

          Learn React

        </a>

      </header>

    </div>

  );

}

export default App;

Mesma coisa com o componente de classe

import logo from './logo.svg';

import './App.css';

import { Component } from 'react';

class App extends Component {

  render() {

    return (

            <div className="App">

              <header className="App-header">

                <img src={logo} className="App-logo" alt="logo" />

                <p>

                  Edit <code>src/App.js</code> and save to reload.

                </p>

                <a

                  className="App-link"

                  href="https://reactjs.org"

                  target="\_blank"

                  rel="noopener noreferrer"

                >

                  Learn React

                </a>

              </header>

            </div>

    );

  }

}

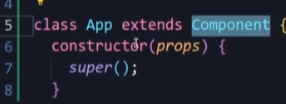
export default App;

Nestes casos não temos estados ou seja, stateless.

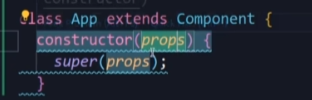
Componentes funcionais e componentes de classe

O estado que vai dizer para o react como o agir.

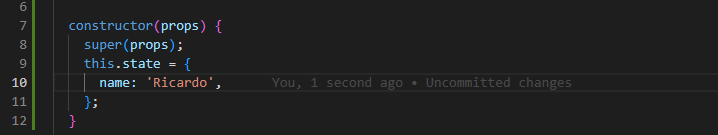
Vamos chamar um constructor na classe e sempre que chamamos um construtor ele recebe props e se a classe extend Componente vamos chamar super() que é para acessarmos o constructor da classe Component.



Passamos a props como parâmetro



Todo componente react pode ter um estado e este estado é basicamente um objeto que contém os dados do componente



A classe fica assim:

import logo from './logo.svg';

import './App.css';

import { Component } from 'react';

class App extends Component {

  constructor(props) {

    super(props);

    this.state = {

      name: 'Ricardo',

    };

  }

  render() {

const name = this.state.name;

    return (

            <div className="App">

              <header className="App-header">

                <img src={logo} className="App-logo" alt="logo" />

                <p>

                 { name }

                </p>

                <a

                  className="App-link"

                  href="https://reactjs.org"

                  target="\_blank"

                  rel="noopener noreferrer"

                >

                  Learn React

                </a>

              </header>

            </div>

    );

  }

}

export default App;

## *\*\* O que é rel = ”noreferrer”?*

*A tag rel = ”noreferrer” é um atributo HTML especial que pode ser adicionado a uma tag de link (a). Ele evita que as informações de referência sejam transmitidas para o site de destino, removendo as informações de referência do cabeçalho HTTP.*

*Isso significa que no Google Analytics, o tráfego proveniente de links com o atributo rel = ”noreferrer” será exibido como Tráfego direto em vez de Referência.*

*Esta é a aparência do atributo noreferrer na visualização HTML:*

***a href="https://www.exemplo.com"******rel="noreferrer"Link to Exemplo.com/a***

## *O que é rel = ”noopener”?*

*rel = ”noopener” é um atributo HTML que pode ser adicionado a links externos. Impede que a página de abertura obtenha qualquer tipo de acesso à página original.*

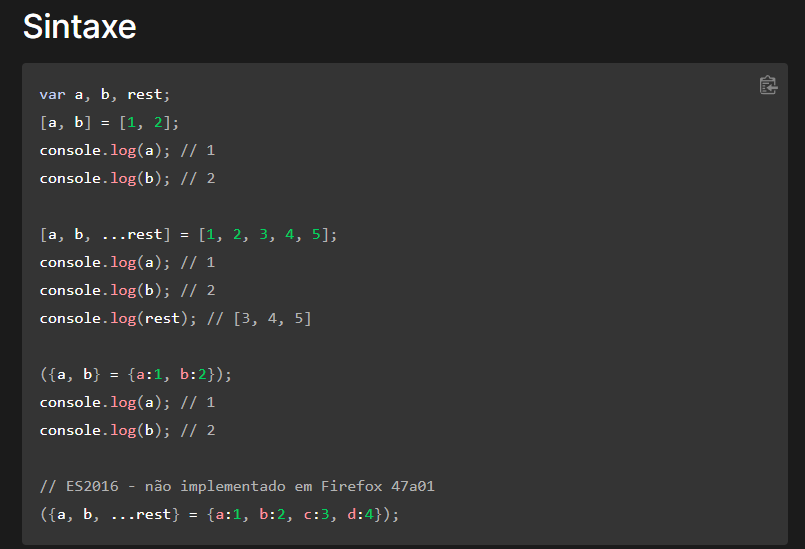
*Aqui está um exemplo de um link com a tag rel = ”noopener”:*

***a href="https://www.exemplo.com****"****rel="noopener"Link to Exemplo.com/a***

*Isso é adicionado automaticamente pelo WordPress em todos os links externos que abrem em uma nova guia por motivos de segurança e é recomendado que você o mantenha.*

*Se você não estiver no WordPress, é recomendável adicionar rel = ”noopener” a todos os seus links externos que abrem em uma nova guia.*

Podemos também usar o destructuring (atribuição via desestruturação):



Outros exemplos:<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment>

Então o:

const name = this.state.name;

Vira:

const { name } = this.state;

com o destructuring.

Evento sintético:

É alguma coisa que acontece na página como uma ação.

Podemos colocar uma ação no nome, por exemplo. Vamos colocar um onclick que chama uma função arrow (arrow function) que vai chamar o console a cada click:

<p onClick={ () => console.log ('<p> Clicado')  }>

            {name}

</p>

Porém vamos chamar um método no click para melhorar a organização da classe:



Podemos retornar o próprio nome na função,

  handlePClick() {

    //const name = this.state.name;

    const { name } = this.state;

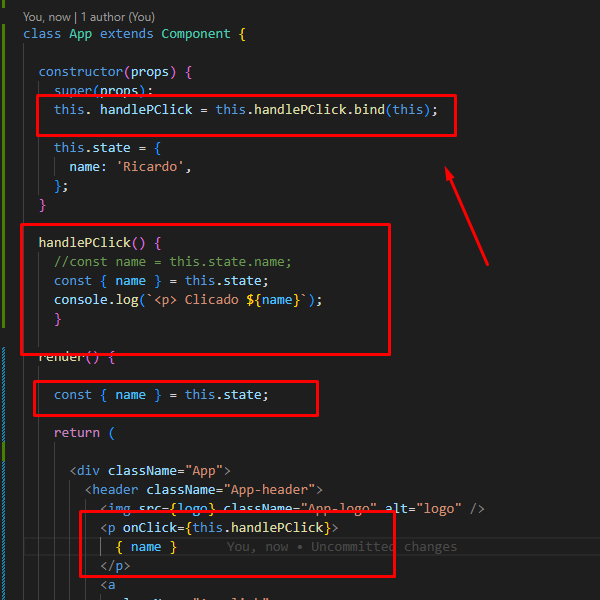
    console.log(`<p> Clicado ${name}`);

    }

*\*usamos template string no console.log*

Porém não conseguiremos pegar o this porque o método não vai fazer o bind (ligar) e o estado vai ficar indefinido.

Precisamos fazer manualmente criando uma bound (vinculada) function



Basicamente estamos falando que este atributo recebe o método e que ele vai receber o this no componente.

2) Mudança de estado.

Agora queremos alterar o estado (o nome neste caso) para outro ao clicarmos no nome.

Não podemos alterar o atributo no state diretamente então vamos usar o setState:

handlePClick() {

    this.setState({ name: 'Schneider' });

  }

Então o nome será alterado ao se clicar no nome na tela.

Cada vez que mudar o estado, render vai ser chamado novamente com o estado novo.

3) agora vamos manipular o link via react

Ao invés de fazermos da maneira anterior, vamos usar uma arrow function para não precisarmos fazer o bound com o this, já que a arrow function não tem this e ela busca no elemento pai que é a classe.

  handleAClick = (event) => {

    event.preventDefault(); // não quero que o evento faça o que ele ia fazer antes (abrir o site)

    const { counter } = this.state

    this.setState({ counter: counter + 1 })

  }

E vamos colocar o onclick na ancora mais abaixo:

 onClick={this.handleAClick}

Criamos um contador e vamos somar a cada click no link

O arquivo todo vai estar assim:

import logo from './logo.svg';

import './App.css';

import { Component } from 'react';

class App extends Component {

  constructor(props) {

    super(props);

    this.handlePClick = this.handlePClick.bind(this);

    this.state = {

      name: 'Ricardo',

      counter : 0

    };

  }

  handlePClick() {

    this.setState({ name: 'Schneider' });

  }

  handleAClick = (event) => {

    event.preventDefault(); // não quero que o evento faça o que ele ia fazer antes (abrir o site)

    const { counter } = this.state

    this.setState({ counter: counter + 1 })

  }

  render() {

    const { name, counter } = this.state;

    return (

      <div className="App">

        <header className="App-header">

          <img src={logo} className="App-logo" alt="logo" />

          <p onClick={this.handlePClick}>

            {name} {counter}

          </p>

          <a

          onClick={this.handleAClick}

            className="App-link"

            href="https://reactjs.org"

            target="\_blank"

            rel="noopener noreferrer"

          >

            Este é o link

          </a>

        </header>

      </div>

    );

  }

}

export default App;

Não esquecendo que podemos utilizar com a arrow function o método handlePClick também.

Só não podemos utilizar arrow functions com alguns métodos que são chamados de LyfeCycleMetodes que será visto posteriormente.

4) Para eliminarmos o método constructor e reduzir o código podemos utilizar algo chamado de class fields *https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes/Public\_class\_fields* que permitem criar atributos na classe sem utilizar construtor.

Então o que era assim:

  constructor(props) {

    super(props);

    this.state = {

      name: 'Ricardo',

      counter: 0

    };

  }

vai ficar apenas com os states:

   state = {

      name: 'Ricardo',

      counter: 0

    };

Posteriormente vamos ver funções com hooks direto sem classe que eliminarão muito código

Link do projeto inicial do professor:

<https://github.com/luizomf/curso-reactjs-nextjs-project-1>