Tecnologias Para Desenvolvimento Web

UNIDADE 03

Componentes em classe e uso de State

Componentes em classe e uso de State

Como já aprendemos na semana passada, os componentes em React, permitem você dividir a UI (interface do usuário) em partes independentes, reutilizáveis e pensar em cada parte isoladamente. Conceitualmente, componentes são como funções JavaScript, que aceitam entradas arbitrárias (chamadas “props”) e retornam elementos React que descrevem o que deve aparecer na tela [1]. Porém, existem componentes mais complexos, como é o caso dos componentes criados a partir de classes. Estes componentes possuem **métodos do ciclo de vida** que podem sobrescrever para executar determinado código em momentos particulares do processo.

Entre os diversos métodos existentes em um componente do tipo classe, temos o **render()**. Ele é o único método obrigatório em um componente do tipo classe. Quando o render() é chamado, ele examina o **this.props** e o **this.state** e retorna elementos React, tipicamente criados via JSX. A Figura 1 demonstra como é a construção de um componente do tipo classe.

Figura 1. Componente do tipo classe

```
class Conteudo extends Component{
    render(){
        return (
            <div>
                <h1> {this.props.nome} {this.props.sobrenome}</h1>
            </div>
        )
    }
}
```

Fonte: o autor, 2021.

Além do render(), que renderiza os componentes, também temos o **constructor** (construtor), que é chamado antes que o componente seja montado, como pode ser visto no exemplo da Figura 2.

Figura 2. Constructor

```
class App extends Component{

    constructor(props){
        super(props);
    }

    render(){
        return (
            <div>

            </div>
        )
    }
}
```

Fonte: o autor, 2021.

Quando estiver implementando o constructor para uma subclasse de React.Component, deve-se chamar super(props) antes de qualquer outra coisa. Caso contrário this.props será indefinido no construtor, o que pode levar a bugs.

Em um componente do tipo classe, também temos o famoso “state”. Ele é um gerenciador que existe dentro do componente, como se fosse uma área para declarar variáveis dentro de uma classe, como atributos, como pode ser visto no exemplo da Figura 3.

Figura 3. State

Fonte: o autor, 2021.

Com o state, podemos inicializar diversos tipos de variáveis dentro do componente e podemos também, atribuir novos valores, por exemplo: ao clicar em um botão na tela, podemos aumentar ou diminuir o valor de uma variável inteira dentro da state. Isso é basicamente mudar o estado do componente, pois se esta variável estiver sendo utilizada dentro do return do render(), este valor será modificado automaticamente.

Mas como podemos criar um botão e fazer com que o clique deste botão mude o valor de uma variável da state? Isso é bem simples na verdade. Mas vamos demonstrar o passo a passo de como podemos fazer isso!

Primeiro vamos declarar uma variável na state chamada ***contador*** e inicializá-la com o valor zero (Figura 4).

Figura 4. Declaração de variável na state

Fonte: o autor, 2021.

Agora precisamos criar um botão para que o usuário possa clicar em algo. Sendo assim, vamos criar a tag button para que o componente possa renderizá-la na página (Figura 5).

Figura 5. Criação do button

Fonte: o autor, 2021.

Se clicarmos no botão apenas como ele está, não irá executar nenhum método para que possamos aplicar nossa lógica para aumentar o valor da variável da state, sendo assim, precisamos criar o atributo ***onClick*** e a partir dele, executar um novo método chamado **aumentar()**, que ainda vamos criar dentro de nosso componente (Figura 6).

Figura 6. Chamada do método no botão

```
<button onClick={this.aumentar}>+</button>
```

Fonte: o autor, 2021.

Agora, devemos criar nosso método chamado **aumentar()**, como pode ser visualizado na Figura 7.

Figura 7. Criação do método aumentar()

```
aumentar(){
  console.log("aumentou");
}
```

Fonte: o autor, 2021.

Esse método dentro do componente executará a função, mas não atualizará o componente na tela. Para isso, é necessário atualizar a state com o método setState, que é nativo do React. O método setState() gera uma atualização do objeto state de um componente. Quando o state muda, o componente responde renderizando novamente na tela automaticamente, ou seja, toda vez que executamos o método setState(), o componente é atualizado na tela, sem a necessidade de atualizarmos pelo browser ou apertar F5.

Desta forma, precisamos criar nossa lógica para aumentar o valor da variável da state e após, atualizar o estado do componente com o método setState (Figura 8).

Figura 8. Lógica do método aumentar()

Fonte: o autor, 2021.

O que está sendo feito na Figura 8, é que estamos obtendo o estado do comente para uma variável também chamada **state**, e dentro dela é que temos a variável **contador**. Assim, modificamos o valor do contador e aplicamos um novo estado para o componente. Porém, ainda temos uma última regrinha: todo método que atualiza o estado do componente, deve ser feito um bind deste método no construtor do componente, para que o componente saiba qual método poderá manipular o estado do componente. O método **bind**() cria uma nova **função**, quando invocado, tem o ***this*** configurado para o valor fornecido. A Figura 9 demonstra como o bind do método aumentar é aplicado ao construtor do componente.

Figura 9. Bind do método do componente

Fonte: o autor, 2021.

Desta forma, podemos construir um componente que tenha métodos, eventos, lógica e reconstrução do estado, apenas utilizado um componente do tipo classe.

CONCLUSÃO

Nesta unidade, foi apresentado o conceito de componentes em classe e estado do componente. Estes dois conceitos são essenciais para o andamento do nosso conteúdo e entendimento da construção de componentes mais avançados.

REFERÊNCIAS

[1] React. ***React – Uma biblioteca JavaScript para criar interfaces de usuário****. Disponível em: https://pt-br.reactjs.org/. Acessado em: 12 de outubro de 2021.*