Tecnologias Para Desenvolvimento Web

UNIDADE 04

Array, imagem e formulário

Array, imagem e formulário

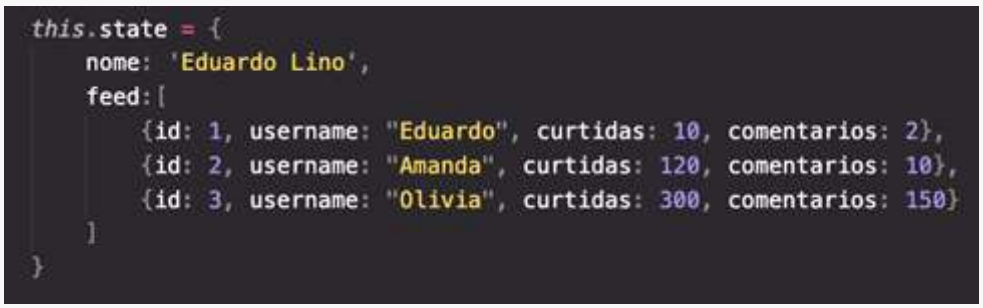
O array que vamos utilizar aqui é basicamente o mesmo visto na primeira semana de aula, a única diferença é que será desenvolvida junto com a sintaxe das aplicações em React. Sendo assim, um array nada mais é que um objeto que permite o armazenamento de uma coleção ordenada de dados indexados sequencialmente a partir de zero.

Porém aqui, com os componentes do tipo classe, não vamos declarar o array em um escopo global do arquivo e sim, dentro do estado do componente, como já vimos na semana passada, quando definidos a variável ***contador*** dentro do estado do

componente.

Como pode ser observado na Figura 1, temos duas variáveis no estado no componente: ***nome*** e ***feed***. A variável ***nome*** é apenas uma String e ***feed*** é um array de objetos com valores já pré-definidos.

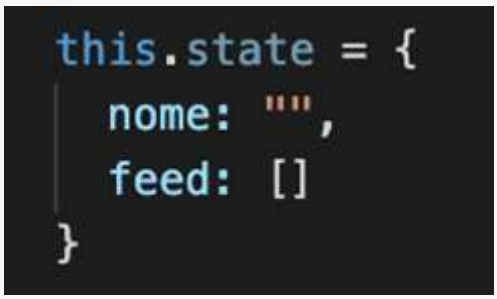
Figura 1. Variáveis de estado do componente



Fonte: o autor, 2021.

Também podemos definir variáveis vazias, como é demonstrado na Figura 2.

Figura 2. Variáveis de estado do componente vazias



Fonte: o autor, 2021.

Para conseguir mostrar uma variável de estado no HTML do componente, precisamos colocá-la, no HTML usando JSX. Para isso, precisamos descrevê-la entre { }, como é mostrado na Figura 3.

Figura 3. Constructor

Fonte: o autor, 2021.

E como sabemos e já vimos na primeira semana, em JavaScript, os arrays não podem ser mostrados na tela apenas colocando-os no HTML, e sim, precisamos percorrer todo o array e mostrar os valores de cada array, pois o array é um objeto. Desta forma, em React, os arrays tem um método chamado **map()**, que basicamente mapeia todos os valores dentro deste array e cria um laço de repetição para que possamos percorrer todo este objeto array. Assim, precisamos apenas descrever a variável ***feed*** (Figura 1), que já possui valores pré-definidos e chamar o método **map().** O método map(), exige que tenhamos uma variável de escopo do método como parâmetro. Esta variável, será os itens de cada posição do array, por exemplo: A cada vez que o laço de repetição map() for executado, a variável item será:

- **Primeira vez que o laço for executado:**

  {id:1, username: “Eduardo”, curtidas: 10, comentários: 2}

- **Segunda vez que o laço for executado:**

  {id:2, username: “Amanda”, curtidas: 120, comentários: 10}

- **Terceira e última vez que o laço for executado:**

  {id:2, username: “Olivia”, curtidas: 300, comentários: 150}

A Figura 4 demonstra como pode ser criado este laço de repetição.

Figura 4. Laço de repetição map() para arrays

```
{
    this.state.feed.map((item) => {
        return(
            <div>
                {item.username}
            </div>
        )
    })
}
```

Fonte: o autor, 2021.

Porém, se você abrir o console da página no browser, poderá perceber que um **warning** está ocorrendo (Figura 5). Lembrando que **warning** não é erro, mas é sempre bom ficar atendo quando algum aparece, pois pode ser que seja muito importante o aviso!

Figura 5. warning

Fonte: o autor, 2021.

Este **warning** indica que, ao criar cada item do **map()**, devemos fazer com que cada um deles tenham uma chave única para diferenciá-los. As chaves ajudam o React a identificar quais itens sofreram alterações, foram adicionados ou removidos. As chaves devem ser atribuídas aos elementos dentro do array para dar uma identidade estável aos elementos, como pode ser visualizado na Figura 6.

Figura 6. Chamada do método no botão

```
{
    this.state.feed.map(item => {
        return(
            <div key={item.id}>
                {item.username}
            </div>
        )
    })
}
```

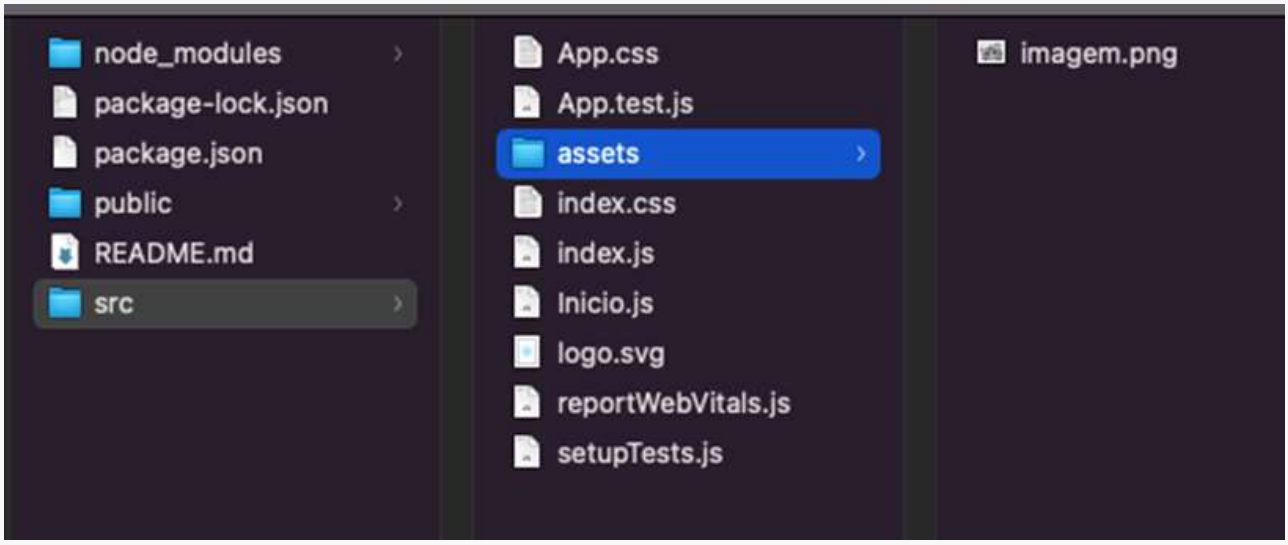
Fonte: o autor, 2021.

Desta forma, atribuímos o ***id*** que faz parte de cada objeto do array como índice. Não é recomendado o uso de índices para chave se a ordem dos itens pode ser alterada. Isso pode impactar de forma negativa o desempenho e poderá causar problemas com o estado do componente.

Como utilizar imagem em React?

Para renderizar uma imagem no HTML do componente, precisamos apenas importar a imagem de algum endereço dentro do projeto React. Sendo assim, podemos baixá-la e salvá-la dentro do projeto. No exemplo da Figura 7, temos uma imagem (imagem.png) que está dentro de uma nova pasta chamada **assets**.

Figura 7. Imagem



Fonte: o autor, 2021.

Agora que temos a imagem dentro do projeto, já podemos importá-la para dentro da nossa aplicação React. Para isso, precisamos importá-la no começo do arquivo que temos o nosso componente, como pode ser visualizado na Figura 8.

Figura 8. Importação de imagem

```
import Imagem from "../src/assets/imagem.png";
```

Fonte: o autor, 2021.

Após fazer a importação, temos o objeto **Imagem** pronto para ser utilizado, como pode ser visualizado na Figura 9.

Figura 9. Uso da imagem

```
return (
  <div>
    <img src={Imagem}></img>
  </div>
);
```

Fonte: o autor, 2021.

Como criar campos de formulário em React?

Formulários são muito comuns em qualquer tipo de aplicação e também precisamos saber como utilizá-los aqui em React. Uma forma de fazermos isso é criando uma variável de estado para cada campo de formulário criado no HTML do componente. No exemplo a seguir, vamos criar um campo de e-mail e fazer com que, ao modificar este campo de e-mail, seu valor deverá estar preenchido dentro da variável de estado, para que possamos obter este valor e fazer o que desejarmos com ele.

Então primeiro vamos criar a variável de estado email, como pode ser visualizado na Figura 10.

Figura 10. Variável de estado, email

Fonte: o autor, 2021.

Em seguida, devemos criar nosso input do tipo email no HTML do componente (Figura 11).

Figura 11. Input do tipo email

Fonte: o autor, 2021.

Em HTML, todo campo de input tem alguns eventos de estado, como é o caso do onChange. Este evento ocorre quando o input em questão tem seu valor modificado. Ou seja, toda a vez que o valor com input for modificado pelo usuário, automaticamente este evento será disparado e podemos fazer com que ele execute um novo método. A Figura 12 mostra como podemos criar um evento de onChange para nosso input email.

Figura 12. Input do tipo email com evento de onChange

```
<input type="text" size="20" name="email" onChange={this.muda} />
```

Fonte: o autor, 2021.

O método **muda** deverá ser criado por nós dentro da nossa classe componente. Assim, toda vez que houver uma mudança no valor deste campo, automaticamente o método **muda** será executado. Mas somente isso, não faz com que o método leve o valor digitado para dentro do método. Precisamos fazer com que um parâmetro seja passado, como é demonstrado na Figura 13.

Figura 13. Input do tipo email com evento de onChange com parâmetro

```
<input type="text" size="20" name="email" onChange={(e) => this.muda(e)} />
```

Fonte: o autor, 2021.

Desta forma, podemos executar nosso código dentro do método **muda()** e obter o valor digitado no input email (Figura 14).

Figura 14. Método muda()

```
muda(event){
    let state = this.state;
    state.email = event.target.value;
    this.setState(state);
    console.log(state.email);
}
```

Fonte: o autor, 2021.

Como já sabemos, o primeiro passo é obter o estado com suas variáveis. Após obter o estado e ter acesso a variável ***email***, precisamos obter o valor digitado do input. Na Figura 12, podemos perceber que foi passado como parâmetro a variável ***e***, e no método muda temos o ***event***, que é mesma variável. Por padrão, geralmente nos inputs são utilizados variáveis com nomes menores apenas para que não fique muito verboso, mas poderia ser utilizado o nome ***event*** nos dois locais. Sendo assim, dentro de ***event***, temos o evento de mudança que foi disparado a partir do onChange e dentro do ***event***, temos acesso ao valor atual digitado, que fica localizado em ***event.target.value***. Após, obter o valor digitado, devemos atualizar a state com este novo valor. Na última linha foi

colocado um **console.log** apenas para teste de visualização no console para verificar se realmente funcionou. Isso é muito importante para quem está começando no universo da programação web.

Desta forma, vamos avançar um pouco mais com as aplicações em React. Com o conteúdo desta semana, podemos criar formulários e imagens, além de trabalhar com arrays, que será muito importante para a continuidade do nosso conteúdo.

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.*