Desenvolvimento para Dispositivos Móveis

Aula 03 - Formulário com React

Material Didático do Instituto Metrópole Digital - IMD

Apresentação

Nesta aula, continuaremos a ver conceitos importantes do *react*. Em especial, veremos como utilizar o *react* para controlar formulários.

Objetivos

- Conhecer como funcionam os *Controlleds Components*
- Conhecer como lidar com múltiplos campos no *react*.

Controlled Components

Link do video da aula: https://youtu.be/SIC3ML3MFsw

Para iniciar a prática, devemos limpar o arquivo '*index.js*', deixando-o apenas com o básico.

Criando formulário

Primeiramente, vamos criar *class component* para controlar um formulário. Acompanhando a aula e observando os conceitos apresentados, implemente o código abaixo:

```
import React from 'react';
import ReactDOM from 'react-dom';

class Form extends React.Component {

  constructor(props) {
    super(props)
    this.state = { value: 'React!' }
    this.handleChance = this.handleChance.bind(this)
  }

  handleChance(event) {
    this.setState({ value: event.target.value })
  }

  render() {
    return (
      <form>
        <input onChange={this.handleChance} value={this.state.value}
/>
```

```
      </form>
    )
  }
}

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

Agora o componente está totalmente sincronizado com a *interface*, porque é possível alterar na *interface*, e isso prova que de fato o estado está sendo alterado.

Testando Submissão

Para visualizar com mais certeza a alteração de estão, incrementaremos uma submissão do valor digitado e apresentaremos esse valor na tela. Acompanhe a aula, efetuando as alterações apresentadas abaixo:

No formulário, adicione um novo *input* abaixo do primeiro e adicione a ação de chamar uma função quando o usuário submeter o formulário:

```
<form onSubmit={this.handleSubmit}>
   <input onChange={this.handleChance} value={this.state.value} />
   <input type="submit" value="Enviar!" />
</form>
```

Depois disso, temos de criar essa função que irá apresentar o valor digitado na tela após a submissão do usuário:

```
handleSubmit(event) {
    alert('Um novo evento foi enviado: ' + this.state.value)
    event.preventDefault()
  }
```

Por fim, é preciso fazer o *bind* da função:

```
this.handleSubmit = this.handleSubmit.bind(this)
```

Controlled Components (Select)

Link do video da aula: https://youtu.be/bnum7rAdhYU

Continuando no assunto de *Controller Component*, vamos ver outro tipo de *input*.

Select

O elemento *select* representa um input que contém um menu de opções formado pelo elemento *option*.

Acompanhando a aula e observando os conceitos apresentados, implemente o código abaixo:

```
import React from 'react';
import ReactDOM from 'react-dom';

class Form extends React.Component {

  constructor(props) {
    super(props)
    this.state = { value: 'branco' }
    this.handleChance = this.handleChance.bind(this)
    this.handleSubmit = this.handleSubmit.bind(this)
  }

  handleChance(event) {
    this.setState({ value: event.target.value })
  }

  handleSubmit(event) {
    alert('Um novo evento foi enviado: ' + this.state.value)
    event.preventDefault()
  }

  render() {
    return (
      <form onSubmit={this.handleSubmit}>
        <select value={this.state.value}
onChange={this.handleChance}>
          <option value="laranja">Laranja</option>
          <option value="branco">Branco</option>
          <option value="verde">Verde</option>
          <option value="amarelo">Amarelo</option>
        </select>
        <input type="submit" value="Enviar!" />
      </form>
```

```
    )
  }
}

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

Lidando com múltiplos campos

Link do video da aula: https://youtu.be/wFcXJGMfB8c

Agora, vamos ver como controlar quando o formulário tem mais de um campo.

Criando vários campos

Atualmente, temos um único campo em nossas práticas, que possui um estado e uma função que lida com as mudanças de estado. Porém, agora vamos criar mais um campo. Daí surge a pergunta: é preciso criar um *state* e uma função *handle* para cada campo? E se você tiver 10 campos? Ou 20 campos?

A resposta é que cada campo precisa sim, ter um estado, não há como fugir disso. Porém, a função que lida com as mudanças pode ser a mesma.

Então, acompanhando a aula, implemente o código abaixo:

```
import React from 'react';
import ReactDOM from 'react-dom';

class Form extends React.Component {

  constructor(props) {
    super(props)
    this.state = { nome: 'Gustavo', cor: 'branco' }
    this.handleChance = this.handleChance.bind(this)
    this.handleSubmit = this.handleSubmit.bind(this)
  }

  handleChance(event) {

    const nameEvt = event.target.name
```

```
    this.setState({ [nameEvt]: event.target.value })

  }

  handleSubmit(event) {

    alert(`O usuário de nome ${this.state.nome} escolheu a cor
${this.state.cor}`)
    event.preventDefault()
  }

  render() {
    return (
      <form onSubmit={this.handleSubmit}>
        <input name="nome" type="text" onChange={this.handleChance}
value={this.state.nome}></input>
        <select name="cor" value={this.state.cor}
onChange={this.handleChance}>
          <option value="laranja">Laranja</option>
          <option value="branco">Branco</option>
          <option value="verde">Verde</option>
          <option value="amarelo">Amarelo</option>
        </select>
        <input type="submit" value="Enviar!" />
      </form>
    )
  }
}

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

Resumo

Nesta aula, vimos como utilizar o *react* para controlar formulários, sejam eles com um único campo ou com múltiplos campos. Ademais, foram vistos conceitos de *inputs* de texto e de seleção.