Desenvolvimento para Dispositivos Móveis

Aula 04 - React Hooks

Material Didático do Instituto Metrópole Digital - IMD

Apresentação

Nesta aula, continuaremos a introdução ao *React*, em particular, veremos como funcionam as *Functions Component*.

Objetivos

- Conhecer como criar componentes baseados em função
- Conhecer as diferenças entre *class components* e *functions components*
- Conhecer como funcionam os *Hooks*

Function Components e Hooks

Link do video da aula: https://youtu.be/lQZ83ub8c0s

Em aulas anteriores já foi mostrado como criar um componente utilizando função. Porém, nesta aula veremos como explorar ainda mais as opções dos *functions components*.

Evolução *functions components*

Os componentes baseados em funções tinham algumas limitações, um exemplo delas seria a manipulação de estados. Então, a partir da versão *16.8* o *react* incrementou ainda mais as *functions components*.

Agora será implementada uma *function component* que armazena um estado e para fazer isso utilizaremos o conceito de *Hooks*.

Hooks é como se fosse uma função que entrega um estado e uma função de atualizar esse estado.

Acompanhe a aula e produza o código abaixo:

```
import React, { useState } from 'react';
import ReactDOM from 'react-dom';

function Toogle(props) {

  const [toogle, setToogle] = useState(true);
  return <button onClick={() => { setToogle(!toogle) }}>{toogle ?
'ON' : 'OFF'}</button>
}
```

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

Para usar os Hooks é importante lembrar de fazer as importações. Veja o exemplo abaixo:

```
import React, {useState } from 'react';
```

É visível a diferença de tamanho do código. E todas as mudanças não alteraram a funcionalidade, mas sim melhoraram o desempenho.

Transformando Class Component em Function Component

Link do video da aula: https://youtu.be/VurdpyrBR2g

A ideia nesta aula é utilizar o código da criação de um formulário (Aula 03) e, ao invés de utilizar *class component*, usarmos *function component*.

Resgatando o formulário

Abaixo, você encontra o código do formulário implementado na Aula 03:

```
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')
);
```

Migrando para *function component*

Agora, acompanhe a aula e implemente no arquivo o seguinte componente:

```
function Form2(props) {

  const [nome, setNome] = useState('Gustavo');
  const [cor, setCor] = useState('branco');

  return (
    <form onSubmit={(event) => {
      alert(`O ${nome} escolheu a cor ${cor} `)
```

```
      event.preventDefault();
    }}>
      <input name="nome" type="text" onChange={(event) =>
setNome(event.target.value)} value={nome}></input>
      <select name="cor" value={cor} onChange={(event) =>
setCor(event.target.value)}>
        <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>
  );
}
```

Para usar os Hooks é importante lembrar de fazer as importações. Veja o exemplo abaixo:

```
import React, {useState } from 'react';
```

Comparando esse componente baseado em função com o componente baseado em classe, é nítida a diferença de tamanho para códigos que têm a mesma finalidade.

Hook UseEffect

Link do video da aula: https://youtu.be/5-no2NxVBEM

Nesse momento continuaremos transformando componentes baseados em classes para *functions components*, para ver os benefícios dessa mudança e fixar melhor esse método.

Resgatando o código

Agora iremos resgatar o componente de *Clock*, o relógio que fizemos. O código encontra-se abaixo:

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

class Clock extends React.Component {
```

```
  constructor(props) {
    super(props)
    this.state = { date: new Date() }
  }

  componentDidMount() {
    this.timerId = setInterval(() => {
      this.setState({ date: new Date() });
    }, 1000);
  }

  componentWillUnmount() {
    clearInterval(this.timerId)
  }

  render() {
    return (
      <div>
        <h1>Hello, world!</h1>
        <h2>It is {this.state.date.toLocaleTimeString()} </h2>
      </div>
    );
  };
}

function App() {
  return (
    <div>
      <Clock />
      <Clock />
    </div>
  );
}

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

Transformando em função

No componente acima foram utilizadas duas funções: *componentDidMount( )* e *componentWillUnmount( )* para atualizar o relógio a cada segundo. Essas funções executam, respectivamente, quando o componente acaba de ser montado e

quando ele é desmontado.

Utilizando *functions components* temos um *Hook* para esse efeito colateral, que é uma alteração feita quando o estado é alterado. Esse *Hook* é denominado *Effect Hook*, que disponibiliza o uso do *useEffect( )*.

Agora, acompanhando a aula, implemente o componente abaixo:

```
function Clock2(props) {

  const [data, setData] = useState(new Date())

  useEffect(() => {
    const id = setInterval(() => setData(new Date(), 1000))
    return () => {
      clearInterval(id)
    }
  }, [])

  return (
    <div>
      <h1>Hello, world!</h1>
      <h2>It is {data.toLocaleTimeString()} </h2>
    </div>
  )
}
```

Para usar os *Hooks* é importante lembrar de fazer as importações. Veja o exemplo abaixo:

```
import React, { useEffect, useState } from 'react';
```

E como esperado, o componente baseado em uma função é bem mais enxuto. Porém, é importante lembrar que ambos funcionam da mesma forma.

Resumo

Nesta aula vimos como funcionam e como implementar *functions components*. Vimos também como manipular estados e efeitos colaterais através dos *Hooks*.

Por fim, para praticar, vimos como transformar alguns componentes baseados em classes em componentes baseados em função.