Desenvolvimento para Dispositivos Móveis

Aula 02 - Fundamentos React (Parte 2)

Material Didático do Instituto Metrópole Digital - IMD

Apresentação

Nesta aula começaremos a ver componentes *react* que armazenam estados e por isso variam sua apresentação de acordo com os valores desses estados.

Objetivos

- Conhecer como funcionam os estados e as propriedades de um componente
- Conhecer como criar componentes que armazenam um estado
- Conhecer como criar componentes que gerenciam seu estado
- Conhecer como criar componentes baseados em classes
- Como utilizar listas no *react*

Tentando encapsular a lógica

Link do video da aula: https://youtu.be/zHQc7cDut4Y

Nesta aula, será implementada uma espécie de relógio, em que o estado será a própria hora, que muda de forma constante.

Criando o relógio

Acompanhando a aula, crie o relógio observando todas as alterações feitas. Ao final, o código produzido deverá estar como se observa abaixo:

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

function Clock(props) {
  const element = (
    <div><h1>Hora: {props.date.toLocaleTimeString()} </h1></div>
  )
  return element;
}

function tick() {
  ReactDOM.render(
    <Clock date={new Date()} />,
    document.getElementById('root')
  );
}
```

```
setInterval(tick, 1000);
```

Essa maneira funciona, porém, ainda não é a maneira correta de se implementar o código. Na próxima aula, será apresentada a maneira certa de fazer isso.

Class components

Link do video da aula: https://youtu.be/ICxeLWKG1aY

Dando sequência ao conteúdo, criaremos um componente que gerencie o seu estado.

Class Component

Até o momento, utilizamos funções para criar componentes *react*. No entanto, para gerenciar o estado é mais vantajoso utilizar o conceito de *Class Component*.

Então, acompanhando a aula e todos os detalhes da criação e das alterações, implemente o código abaixo:

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

class ClockClass extends React.Component {

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

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

  render() {
    return (<div><h1>Hora: {this.state.date.toLocaleTimeString()}
</h1></div>);
  };
}

ReactDOM.render(
```

```
  <ClockClass />,
  document.getElementById('root')
);
```

Limpando o intervalo

Nosso componente está fazendo uma chamada a cada segundo para atualizar o estado. Porém, caso precise remover esse component da tela, é interessante limpar esse intervalo para não gastar recursos à toa.

O *setInterval( )* retorna um *id*. Então, vamos armazenar esse *id*. Na função *componentDidMount( )*:

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

Agora, para limpar o *setInterval( )* , usaremos uma função que é executada quando o componente é desmontado: *componentWillUnmount( )*.

Então, abaixo da função *componentDidMount( )* implemente o código a seguir:

```
componentWillUnmount() {
    clearInterval(this.timerId)
  }
```

Essa alteração não ocasionará mudanças visuais, mas o componente está gerenciando melhor os recursos.

Reagindo ao usuário

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

Até agora criamos componentes que reagem apenas internamente, alterando seu estado. Nesta aula, veremos como criar um componente que altere seu estado a partir de uma ação do usuário.

Criando um botão

Para praticar, criaremos um botão que, ao usuário clicar, o estado será alterado e o

react representará isso de forma visual.

Então, novamente, criaremos um componente do zero. Acompanhando e observando detalhadamente a aula, implemente no 'index.js' o componente abaixo:

```
class Toogle extends React.Component {

  constructor(props) {
    super(props)
    this.state = { isToogleOn: false }
    this.handleClick = this.handleClick.bind(this)
  }

  handleClick() {
    this.setState(prevState => ({
      isToogleOn: !prevState.isToogleOn
    }))
  }

  render() {
    return (
      <button onClick={this.handleClick}>
        {this.state.isToogleOn ? 'ON' : 'OFF'}
      </button>
    );
  }
}
```

Listas e Keys

Link do video da aula: https://youtu.be/7DaHOxrkARE

Já vimos como criar componentes *react* com funções e classes e conhecemos a função das *props* e do *state*. Agora, vamos ver como lidar com listas no *react*.

Exemplo de lista

É muito comum precisarmos listar algo, principalmente listar algo dinâmico. Então, para assimilação do conhecimento, acompanhe a aula implementando o código abaixo:

```
import React from 'react';
```

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

const numbers = [1, 2, 3, 4, 5, 6, 7]

const listItems = numbers.map((number) => {
  return <li>{number}</li>
})

const element = listItems

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Identificador Único

O código acima funcionará, porém, com *warnings* e não é uma boa prática ignorar os avisos do compilador.

O problema é que, ao usar lista, cada elemento precisa ter algo que o identifique de forma única, uma *key* passada como propriedade da *tag*. Existem vários meios de fazer isso, mas o que usaremos aqui é passar o *index* de cada elemento como seu identificador.

Então, no mapeamento dos itens adicione essa chave. Veja abaixo:

```
const listItems = numbers.map((number, index) => {
  return <li key={index}>{number}</li>
})
```

Resumo

Nesta aula vimos a função dos estados e das propriedades de um componente. Vimos também como criar componentes que armazenam e gerenciam um estado. Ademais, foi visto como se trabalhar com listas no *react*, pois muitas vezes precisamos manipular um conjunto de dados e não apenas um dado isolado.