IOS – Instituto de Oportunidade Social

HTML CSS JS 34 - Componentes com Classes

Objetivos

- Criando componentes utilizando classe
- Criando componentes utilizando classe
- Inserindo um novo componente na hierarquia
- Adicionando um estilo a um componente

IOS – Instituto de Oportunidade Social

States

States

Em componentes implementados com classe, utilizamos o conceito de state, ao invés de hooks como em componentes funcionais. Desse modo, o componente criado com classe terá o seu state, que poderá possuir um ou mais campos

```
class NomeComponente extends Component {
  state = {
    campo01: valor01;
    campo02: valor02;
    ...
    campoN: valorN;
  }
  render() {
    return (
      ...
    );
  }
}
```

States

E para acessar esse estado devemos utilizar this:

```
this.state.campoX
```

E para atualizar o estado utilizamos this.detState():

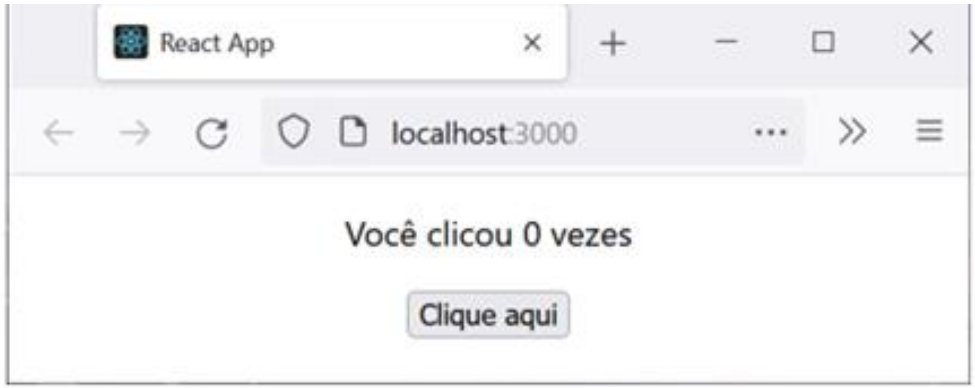
```
this.setState(campoX: novo_valor)
```

IOS – Instituto de Oportunidade Social

Criando componentes utilizando classe

Componente utilizando classe

Vamos fazer o exemplo do tema 01 com o componente utilizando classe que possui um parágrafo e um botão, onde o parágrafo mostra quantas você clicou no botão da interface



Comando para criar aplicação em React

npx create-react-app tema_04_conta_cliques_classe

Componente utilizando classe

```
1.  import { Component } from 'react';
2.
3.  class Exemplo extends Component {
4.      // Declare uma nova variável de estado, a qual chamaremos de "contador"
5.      // e é um campo do objeto state
6.      constructor(props) {
7.          super(props);
8.          this.state = {
9.              contador: 0,
10.         };
11.     }
12.
13.     render() {
14.         return (
15.             <div>
16.                 <p>Você clicou {this.state.contador} vezes</p>
17.                 <button
18.                     onClick={() =>
19.                         this.setState({ contador: this.state.contador + 1 })
20.                     }
21.                 >
22.                     Clique aqui
23.                 </button>
24.             </div>
25.         );
26.     }
27. }
28.
29. export default Exemplo;
```

Componente utilizando classe

Importante: Note, também, que você só pode ter um elemento React pai dentro do método return. Desse modo precisamos colocar o elemento para envolver os outros dois elementos, parágrafo e botão. Dessa forma, a aplicação funcionará corretamente."

Isso não é permitido

```
return (
    <p>Você clicou {this.state.contador} vezes</p>
    <button
        onClick={() =>
            this.setState({ contador: this.state.contador + 1 })
        }
    >
        Clique aqui
    </button>
);
```

Isso está correto

```
return (
    <div>
        <p>Você clicou {this.state.contador} vezes</p>
        <button
            onClick={() =>
                this.setState({ contador: this.state.contador + 1 })
            }
        >
            Clique aqui
        </button>
    </div>
);
```

Componente utilizando classe

Na linha 3 do código mostrado, temos a instrução para importar o componente filho no componente pai App e, desse modo, ele ser reconhecido como um componente válido. Caso você esqueça de colocar essa instrução, a aplicação acusará erro na linha 9.

```
1.  import './App.css';
2.  import { Component } from 'react';
3.  import Exemplo from './Components/Exemplo';
4.
5.  export class App extends Component {
6.      render() {
7.          return (
8.              <div className="App">
9.                  <Exemplo />
10.             </div>
11.         );
12.     }
13. }
14.
15. export default App;
```

Componente utilizando classe

A linha 9 contém o nosso componente criado, indicando o local (ordem) que ele deverá ser renderizado. Note que devemos sempre usar a notação:

Abre o componente com o símbolo de menor que	Nome do componente	Espaço	Fechar o componente com barra e o sinal de maior que
<Exemplo />			

IOS – Instituto de Oportunidade Social

Inserindo componentes na hierarquia

Inserindo componentes na hierarquia

Os componentes do React são elementos **autocontidos** que você **reutiliza** por toda uma página. Ao criar pequenos trechos de código focados, você move e reutiliza esses trechos à medida que seu aplicativo cresce. O principal aqui é que eles são autocontidos e focados, permitindo que você **separe o código** em partes lógicas.

Inserindo componentes na hierarquia

Pacote PropTypes

Ao utilizar o pacote PropTypes em componentes implementados com classe não precisamos fazer como componentes funcionais e indicar que estamos esperamos uma propriedade basta colocar o acesso, Ex:

```
{this.props.nome}
```

Inserindo componentes na hierarquia

Exemplo de classe com PropTypes:

```
import { Component } from 'react';
import PropTypes from 'prop-types';

export class Header extends Component {
    render() {
        return (
            <div>
                <h1>Olá, {this.props.nome}, seja bem-vindo!</h1>
            </div>
        );
    }
}
```

Inserindo componentes na hierarquia

Valor padrão para a propriedade:

```
Header.defaultProps = {
    nome: 'Nome padrão',
};
```

Validação do tipo de dado da propriedade:

```
Header.propTypes = {
    nome: PropTypes.string,
};
```

IOS – Instituto de Oportunidade Social

Adicionando estilo ao componente

Adicionando estilo ao componente

Formas de Estilização

Existem várias formas de configurar estilos em um componente React: **inline**, pelo arquivo **App.css** ou **index.css**, através de Bootstrap, etc. Vamos iniciar com um primeiro exemplo de aplicar estilos CSS **inline** no componente.

Adicionando estilo ao componente

camelCase x JSX

Para formatar o estilo, iremos utilizar algumas propriedades que devem ser separadas por vírgula e que no **CSS** são conhecidas por: **border-bottom**, **background-color** e etc. No **JSX** são utilizadas com **camelCase** ficando como: **borderBottom**, **backgroundColor** e etc.

Adicionando estilo ao componente

Exemplo do estilo em JSX:

```
const estilo = {
    color: 'red',
    borderBottom: 'black solid 2px',
    backgroundColor: '#E6E6E6',
};

…

<h1 style={estilo}>Olá, {this.props.nome}, seja bem-vindo!</h1>
```

IOS – Instituto de Oportunidade Social

Vamos Praticar - Parte 2

Vamos Praticar - Parte 2

Apostila de React:

Tema_04_Componentes_Classes

Páginas 7 a 10

OBS: Acompanhar o passo a passo com o instrutor

IOS – Instituto de Oportunidade Social

Exercício

Exercício

Construir um novo App do React onde seu componente principal deverá ter no mínimo 3 elementos estruturais do HTML (Ex: h2, section, footer) realizando personalizações de CSS no formato **inline** (direto em JSX, Ex: **color**, **borderBottom**, **backgroundColor**, ...), reutilizar o componente **Header.jsx** na chamada da página principal, fazer o Build e publicar no GitHub.