Curso Presencial Programação Fullstack

Aula 04

Prof. MSc. Kelson | Senior Software Engineer

PROFKELSON.DEV

BORA CODAR?

Cronograma

01

React JS

02

03

PROFKELSON.DEV

BORA CODAR?

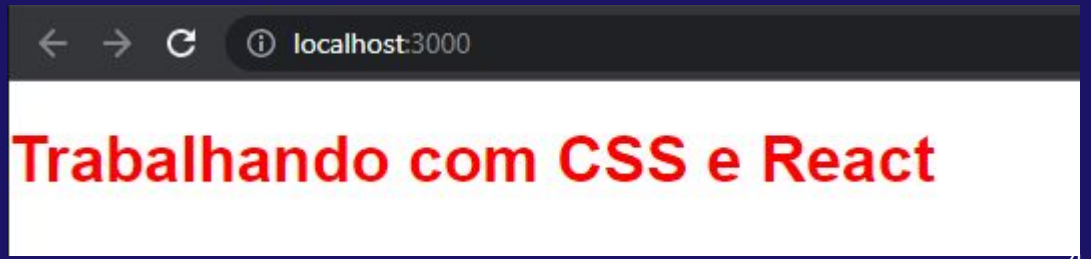
CSS Global

- Estilizar elementos em comum ou fazer reset no CSS
  - Index.css (src)
    - Os estilos globais vão estar nesse arquivo.

PROFKELSON.DEV
BORA CODAR?

CSS Global

- Estilizar elementos em comum ou fazer reset no CSS
  - Index.css (src)
    - Os estilos globais vão estar nesse arquivo.



PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (1) [COM O PROF]

- Crie um estilo global que contenha:
- A cor de fundo do corpo deve ser #d77d7d.
- O tamanho da fonte padrão deve ser definido como 50px.
- Todas as imagens devem ter uma borda sólida de 1px na cor #ccc.
- Crie um componente que mostre um parágrafo <p> e imagem para verificar se o CSS surtiu efeito.

PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (1_1) [SOZINHO(A)]

- Crie um estilo global que contenha:
- A cor de fundo do corpo deve ser #7dd7b9.
- O tamanho da fonte padrão deve ser definido como 20px.
- Todos os botões (<button>) devem ter um fundo #4d79ff, cor do texto #ffffff, e sem borda (border: none;).
- Os links (<a>) devem ter a cor #ff4d4d e não devem ter sublinhado (text-decoration: none;).
- Implemente um componente para testar seu css global.

PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (1_2) [SOZINHO(A)]

- Crie um estilo global que contenha:
- A cor de fundo do corpo deve ser #f0f0f0.
- O tamanho da fonte padrão deve ser definido como 16px.
- Todos os inputs (<input>) devem ter uma borda arredondada de 5px, uma borda de 1px sólida na cor #bbb, e um padding de 10px.
- Todos os elementos de cabeçalho (<h1>, <h2>, <h3>, etc.) devem ter a cor #3b3b3b e o texto centralizado (text-align: center).

PROFKELSON.DEV
BORA CODAR?

CSS inline

- Inline do React é bem parecido com o do CSS
- Podemos encontrar em alguns projetos o atributo style diretamente em algum componente.
- É preferível que optemos por outra maneira, inline dificulta a manutenção.
- Detalhes:
- {{ }} (duas chaves para abrir e fechar o style.
- camelCase para separar as palavras compostas, no exemplo ao lado “background-color” virou “backgroundColor”

```css
const minhaDiv = (
  <div
    style={{
      backgroundColor: 'blue',
      color: 'white'
    }}
  >
    Olá, mundo!
  </div>
);
```

PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (2) [COM O PROF]

- Adicione um estilo inline usando CSS em um componente React.
- O estilo deve ter o seguinte:
  - A cor do texto deve ser vermelha (#ff0000).
  - O fundo deve ser azul (#0000ff).
  - A largura deve ser de 200 pixels.
  - O alinhamento do texto deve ser centralizado.

PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (2_1) [SOZINHO(A)]

- Adicione um estilo inline usando CSS em um componente React.
- O estilo deve ser declarado numa “const” e só depois ser chamado no style do componente.
  - const style = ...
  - ...
  -
- O estilo deve ter o seguinte:
  - A cor do texto deve ser verde (#008000).
  - O fundo deve ser amarelo (#ffff00).
  - A altura deve ser de 100px.
  - A fonte deve ser em negrito.

PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (2_2) [SOZINHO(A)]

- Adicione um estilo inline usando CSS em um componente React.
- O estilo deve ter o seguinte:
  - A cor do texto deve ser laranja (#ffa500).
  - O fundo deve ser cinza (#808080).
  - A altura deve ser de 150 pixels.
  - A margem deve ser de 20 pixels.

PROFKELSON.DEV
BORA CODAR?

CSS inline dinâmico

- Estilo baseado em uma condicional
- No exemplo ao lado temos um caso com o if ternário que aprendemos
- Dependendo do valor mudou a regra de exibição de um “background”

```css
const isDarkMode = true;
const minhaDiv = (
  <div
    style={{
      backgroundColor: isDarkMode ? 'black' : 'white',
      color: isDarkMode ? 'white' : 'black'
    }}
  >
    Olá, mundo!
  </div>
);
```

PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (3) [COM O PROF]

- Adicione um estilo inline dinâmico usando CSS em um componente React.
  - Use um if-else ou operador ternário para alterar o estilo com base em uma condição.
  - Se a condição for verdadeira:
    - A cor do texto deve ser laranja (#ffa500).
    - O fundo deve ser cinza (#808080).
    - A altura deve ser de 150 pixels.
    - A margem deve ser de 20 pixels.
  - Caso contrário:
    - A cor do texto deve ser verde (#008000).
    - O fundo deve ser amarelo (#ffff00).
    - A altura deve ser de 100 pixels.
    - A margem deve ser de 10 pixels.

PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (3_1) [SOZINHO(A)]

- Adicione um estilo inline dinâmico usando CSS em um componente React.
- Use um if-else ou operador ternário para alterar o estilo com base em uma condição.
- Se a condição for verdadeira:
  - A cor do texto deve ser azul (#0000ff).
  - O fundo deve ser rosa (#ffc0cb).
  - A largura deve ser de 300 pixels.
  - O padding deve ser de 15 pixels.
- Caso contrário:
  - A cor do texto deve ser preto (#000000).
  - O fundo deve ser branco (#ffffff).
  - A largura deve ser de 250 pixels.
  - O padding deve ser de 5 pixels.

PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (3_2) [SOZINHO(A)]

- Adicione um estilo inline dinâmico usando CSS em um componente React.
- Use uma condição mais elaborada para alterar o estilo com base em uma combinação de duas variáveis.
- Se ambas as variáveis isPrimary e isLarge forem verdadeiras:
  - A cor do texto deve ser branco (#ffffff).
  - O fundo deve ser azul (#0000ff).
  - A largura deve ser de 400 pixels.
  - O padding deve ser de 20 pixels.
  - Se isPrimary for verdadeira e isLarge for falsa:
  - A cor do texto deve ser branco (#ffffff).
  - O fundo deve ser verde (#008000).
  - A largura deve ser de 200 pixels.
  - O padding deve ser de 10 pixels.
  - Se isPrimary for falsa e isLarge for verdadeira:
  - A cor do texto deve ser preto (#000000).
  - O fundo deve ser amarelo (#ffff00).
  - A largura deve ser de 300 pixels.
  - O padding deve ser de 15 pixels.
- Se ambas as variáveis forem falsas:
  - A cor do texto deve ser preto (#000000).
  - O fundo deve ser cinza (#808080).
  - A largura deve ser de 150 pixels.
  - O padding deve ser de 5 pixels.

CSS de Componente

- Para o componente em específico
- Mesmo nome do componente, importando no componente.
- É criado um exemplo com (App.jsx e App.css)
- O CSS pode vazar para outros componentes.
  - Tem que ter cuidado!
  - Ele pode vazar, devemos usar regras mais restritas.
  - Usar className da tag para fechar o escopo do CSS somente na tag do componente desejado.

CSS de Componente

- Para o componente em específico
- Mesmo nome do componente, importando no componente.
- É criado um exemplo com (App.jsx e App.css)
- O CSS pode vazar para outros componentes.
  - Tem que ter cuidado!
  - Ele pode vazar, devemos usar regras mais restritas.
  - Usar className da tag para fechar o escopo do CSS somente na tag do componente desejado.

CssComponente.js U | CssComponente.css U

reacting_1403 > src > components > AulasCss > CssComponente.js > CssComponente

```jsx
import './CssComponente.css'

const CssComponente = () => {
  return (
    <div>
        <p className="p-CssComponente">Um parágrafo no CSS de Componente...</p>
    </div>
  )
}

export default CssComponente
```

CssComponente.js U | CssComponente.css U

reacting_1403 > src > components > AulasCss > CssComponente.css > .p-CssComponente

```css
p {
    color: black;
    background-color: cornflowerblue;
}

.p-CssComponente {
    color: pink;
    background-color: cornflowerblue;
}
```

PROFKELSON.DEV
BORA CODAR?

CSS dinâmico (com classes)

- Poderíamos fazer esse dinamismo sem a obrigatoriedade de ser em inline
- Mais interessante.
- Por exemplo, criando as classes CSS e chamando a classe a ser usada de acordo com uma condicional

css

```css
.light-mode {
  background-color: white;
  color: black;
}

.dark-mode {
  background-color: black;
  color: white;
}
```

javascript

```javascript
import React from 'react';
import './styles.css';

const minhaDiv = ({ isDarkMode }) => {
  const classe = isDarkMode ? 'dark-mode' : 'light-mode';
  return <div className={classe}>Olá, mundo!</div>;
};
```

CSS Module

- Arquivos CSS modulares.
- CSS que corresponde a cada componente.
- O estilo fica totalmente contido no escopo do componente.
- Evitando mais ainda vazar para outros componentes.
- O arquivo criado com o final ".module.css"

javascript

```javascript
import React from 'react';
import styles from './MeuComponente.module.css';

const MeuComponente = () => {
  return (
    <div className={styles.myClass}>
      Olá, mundo!
    </div>
  );
};

export default MeuComponente;
```

css

```css
.myClass {
  color: red;
}

.myOtherClass {
  background-color: blue;
}
```

PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (4) [COM O PROF]

- Crie um CSS Modularizado para um componente chamado Campanha.
- Esse componente exibe na tela uma mensagem de acordo com o mês.
  - Essa frase deve ser exibida na cor preta.
- A cor de fundo de uma tarja (pode ser uma div) e a mensagem (dentro da tarja) devem mudar de acordo com o mês que passamos como prop (string) para o componente filho.
  - Setembro -> cor: amarelo, mensagem: Prevenção ao suicídio.
  - Outubro -> cor: rosa, mensagem: Conscientização sobre o câncer de mama.
  - Novembro -> cor: azul, mensagem: Prevenção e combate ao câncer de próstata.

PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (4_1) [SOZINHO(A)]

- Crie um CSS Modularizado para um componente chamado EstacaoAno.
- Esse componente exibe uma mensagem com uma cor de fundo que varia de acordo com a estação do ano passada como prop.
- O componente deve receber uma prop estacao (string).
- A mensagem deve ser exibida em uma div com cor de fundo específica para cada estação:
    - Verão -> cor: laranja, mensagem: "Tempo de praia e sol!"
    - Outono -> cor: marrom, mensagem: "Folhas caindo, outono chegando!"
    - Inverno -> cor: cinza, mensagem: "Época de frio e aconchego!"
    - Primavera -> cor: verde, mensagem: "Flores e cores, é primavera!"

PROFKELSON.DEV
BORA CODAR?

Formulários

- Tag <form>
- Labels contém: htmlFor, com o valor do name do input
- O mais recomendado é utilizar a tag <label> ... </label> entre os inputs
- Não utiliza action, o processamento é assíncrono.

```jsx
import React from 'react';

function Formulario() {
  return (
    <div>
      <form>
        <div>
          <label>
            Nome:
            <input type="text" name="nome" />
          </label>
        </div>
        <div>
          <label>
            E-mail:
            <input type="email" name="email" />
          </label>
        </div>
      </form>
    </div>
  );
}

export default Formulario;
```

Formulários

- Com o hook useState
- Armazenar na variável de estado e usar o “set” para mudar o valor.
- Criar uma função para alterar o valor no evento onChange.
- Pensando em: Enviar dados para um back-end -> banco de dados. Por exemplo.

```jsx
import { useState } from 'react'

const ManipularValores = () => {

    const [nome, setNome] = useState()
    const [email, setEmail] = useState()

    const handleName = (e) => {
        setNome(e.target.value)
    }

    const handleEmail = (e) => {
        setEmail(e.target.email)
    }

```

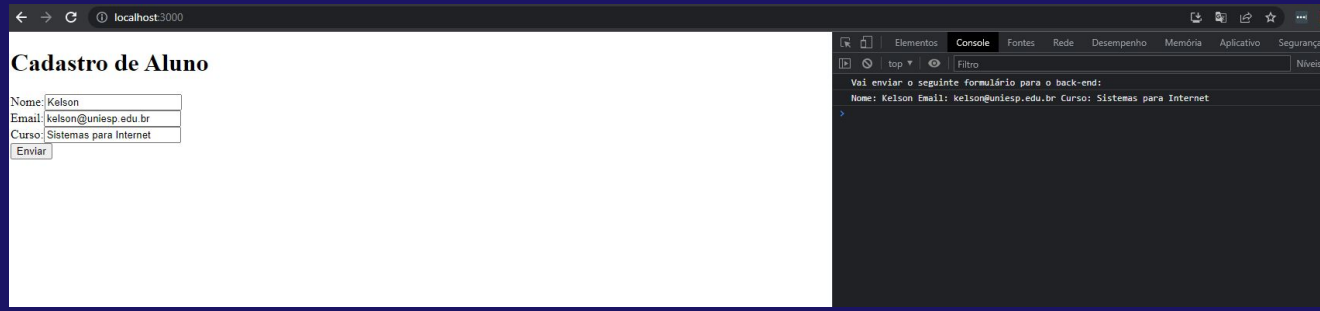
```jsx
  return (
    <div>
        <form>
            <label>
                Nome:
                <input name='nome' type='text' onChange={handleName} />
            </label>
            <br/>
            <label>
                Email:
                <input name='email' type='text' onChange={handleEmail} />
            </label>
            <br/>
            <button>Enviar</button>
        </form>
    </div>
  )
}
```

PROFKELSON.DEV
BORA CODAR?

Formulários

- onSubmit dentro da tag form.
- Criamos uma função para receber o processamento do onSubmit, geralmente damos o nome de "handleSubmit".

```
21    const handleSubmit = (e) => {
22        e.preventDefault()
23        console.log("Vai enviar o seguinte formulário para o back-end: ")
24        console.log(`Nome: ${nome} Email: ${email} Curso: ${curso}`)
25    }
26
27  return (
28    <div>
29        <h1>Cadastro de Aluno</h1>
30        <form onSubmit={handleSubmit}>
```



PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (5) [COM O PROF]

- Crie um componente chamado FormularioDeContato
- Neste componente crie um formulário com os campos:
  - Nome
  - Contato
  - Mensagem
- Ao enviar esse formulário a função responsável pelo submit recebe esses dados em um objeto literal e converte para JSON.
- Sintaxe: let jsonToSend = JSON.stringify(objetoLiteral)
- Ao fim do submit imprima através de console.log a mensagem: O seguinte JSON será enviado via HTTP POST para o back-end: ${jsonToSend}

PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (5_1) [SOZINHO(A)]

- Crie um componente chamado FormularioDeRegistro.
- Neste componente, crie um formulário com os campos:
  - Username
  - Email
  - Password
  - Confirm Password
  - Endereço
  - Telefone
- Ao enviar esse formulário, a função responsável pelo submit deve receber esses dados em um objeto literal e convertê-los para JSON.
- Sintaxe: let jsonToSend = JSON.stringify(objetoLiteral)
- Ao fim do submit, imprima através de console.log a mensagem: O seguinte JSON será enviado via HTTP POST para o back-end: ${jsonToSend}

orkut

PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (6) [COM O PROF]

- Crie um componente chamado SimuladorDeScrap que simule o envio de scraps na antiga rede social Orkut.
- Neste componente, crie um formulário com os campos:
  - Para
  - Mensagem
  - Botões "Postar" e "Cancelar"
- Use CSS Modules no componente para simular as cores e o estilo do Orkut, conforme a imagem fornecida.
- O componente deve ter as seguintes características: (o prof vai disponibilizar o css pronto!)
  - O background deve envolver a página inteira.
  - A logo do Orkut deve ser exibida no topo da página, redimensionada para ficar pequena.
  - O menu deve conter os links "Home", "Perfil", "Scraps", "Comunidades" e "Aplicativos", cada um envolto por um retângulo com uma cor de fundo específica.
  - Os inputs para os campos "Para" e "Mensagem" devem ser um pouco menos largos (90% do container).
  - Ao enviar o formulário, a mensagem deve ser exibida na tela abaixo do formulário, em vez de mostrar o envio do JSON no console.
  - Ao fim do submit, exiba o scrap postado na tela com as seguintes informações:
    - Para: (destinatário do scrap)
    - Mensagem: (conteúdo do scrap)

orkut

PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (6) [COM O PROF]