**ANOTAÇÕES - REACT - ALURA**

**DOWNLOADS NECESSÁRIOS**

Visual Studio Code - v.1.100.1

NodeJS - v.22.12 / v.22.15

**PASSOS INICIAIS**

Pasta alura-books

Abrir pelo terminal do VS Code

*npx create-react-app .*

*npm start* - roda a aplicação

No terminal é mostrado o endereço do localhost onde está sendo executado.

CTRL+c - para a aplicação

**Sobre os arquivos da estrutura base**

package.json - configurações básicas - nome do projeto, versão, dependências, ....

src/ - códigos da aplicação(app.css, app.js)

**COMPONENTES NO REACT**

Componentes são funções JS cujo retorno é um HTML, logo, podem ser criados componentes que assumem o papel de trechos do front-end, tais como imagens de cabeçalho, parágrafos, divs, ...., qualquer parte do front que possa ser representada por um trecho HTML pode se tornar um componente.

Com essa facilidade em dividir o código em componentes, também é facilitado a organização. Por exemplo, uma logo da página é um componente Logo que fica na pasta componentes/Logo/index.js

Nesse arquivo pode ser aplicado um CSS único dele, e todo esse componente, já com a estilização, pode ser importado na página App.js

**STYLED COMPONENTS**

Pacote para estilização de componentes. Evitar o uso do CSS puro e adotar a utilização do pacote

Instalação: no terminal vscode - npm install styled-components

Exemplo: Trocar a estilização CSS da classe App.js

Atualmente:

function App() {//FUNÇÃO JS QUE RETORNA UM HTML

  return (

    <div className='App'>

          <Header/>

    </div>

  );

}

Com o CSS

.App{

  width: 100vw;

  height: 100vh;

  background-image: linear-gradient(90deg,#002f52 35%, #326589 165%);

}

li{

  list-style: none;

}

  justify-content: center;

}

Para que isso possa ser feito com o styled components, mudar no arquivo principal (App.js) para

import Header from './componentes/Header';

import styled from 'styled-components';

const AppContainer = styled.div`

  width: 100vw;

  height: 100vh;

  background-image: linear-gradient(90deg,#002f52 35%, #326589 165%);

  li{

  list-style: none;

}

`

function App() {//FUNÇÃO JS QUE RETORNA UM HTML

  return (

    <AppContainer>

          <Header/>

    </AppContainer>

  );

}

Agora não é mais necessário o arquivo CSS e o estilo aplicado ao componente <Header/> está no mesmo arquivo, tudo em JS.

**Isso é mesmo vantajoso? Pq não usar o CSS local? Qual a diferença?**

Parece ser utilizável apenas para situações onde o CSS do componente seja simples o bastante para ser utilizado somente pelo container direto no arquivo js.

**GLOBAL STYLE**

Uma maneira de definir algumas configurações de estilo únicas para todo o projeto. Em **src/index.js**

Importar pacote createGlobalStyle, do styled-compoments e definir dentro dele as configurações globais

import { createGlobalStyle } from 'styled-components';

const GlobalStyle = createGlobalStyle`

body {

  margin: 0;

  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',

    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',

    sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

}

code {

  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',

    monospace;

}

li {

  list-style: none;

}

`

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <React.StrictMode>

    <GlobalStyle />

    <App />

  </React.StrictMode>

);

Excluir o arquivo index.css (dado que suas definições estão no contêiner GlobalStyle).

Agora, o estilo para tags body, code e li são globais no projeto.

\*O li foi retirado da definição de container em App.js

Aplicação de Styled Components para todos os componentes. Em componentes/

IconesHeader - excluir estilos.css

import styled from 'styled-components'

const Icones = styled.ul`

    display: flex;

    align-items: center;

`

const Icone = styled.li`

    margin-right: 40px;

    width: 25px;

    cursor: pointer;

`

const icones = [perfil, sacola];//ÍCONES DO HEADER

function IconesHeader(){

    return(

         <Icones>

            {icones.map((icone) => (

                <Icone>

                    <img src = {icone} alt='icone do cabeçalho'></img>

                </Icone>

          ))}

        </Icones>

    )

}

OpcoesHeader->excluir estilo.css

import styled from 'styled-components'

const Opcoes = styled.ul`

    display: flex;

    align-items: center;

`

//CONTEINER li

const Opcao = styled.li`

    font-size: 16px;

    display: flex;

    justify-content: center;

    align-items: center;

    text-align: center;

    height: 100%;

    padding: 0 5px;

    cursor: pointer;

    min-width: 120px;

`

const textoOpcoes = ['CATEGORIA','FAVORITOS','MINHA ESTANTE'];//OPÇÕES PARA O HEADER

function OpcoesHeader(){

    return(

        <Opcoes>

            { textoOpcoes.map((texto) =>

                (<Opcao>

                    <p>{texto}</p>

                </Opcao>))}

        </Opcoes>

    )

}

Logo-> excluir estilo.css

import styled from 'styled-components'

//CONTAINER PARA DIV

const LogoContainer = styled.div`

    display: flex;

    font-size: 30px;

`

//CONTAINER PARA IMAGEM

const LogoImage = styled.img`

    margin-right: 10px;

`

function Logo() {

    return(

         <LogoContainer>

            <LogoImage

            src={logo}

            alt='logo alura books'

            />

            <p><strong>Alura</strong>Books</p>

        </LogoContainer>

    )

}

**COMPONENTE QUE SÓ GUARDA ESTILO**

No projeto, a função de pesquisa foi feita com o componente Pesquisa, em App.js

Esse componente, especificamente, usa outro componente Input, que só guarda o estilo para tags do tipo input. Agora, dado esse campo de pesquisa, como filtrar o que aparece na tela pelo que está escrito no campo?

**ESTADOS NO REACT**

Estados são maneiras de guardar valores em constantes, usando um pacote específico do react. Esses valores podem ser utilizados para filtrar pesquisas,.....

No exemplo do projeto é usado um estado chamado de textoDigitado, que guarda o valor inserido no campo de pesquisa e mostra ele numa tag p abaixo do campo, sempre que o evento onBlur ocorre, ou seja, sempre que algo é digitado no campo de pesquisa, e a pessoa clica fora dele, esse conteúdo aparece na página.

import { useState } from 'react'; //IMPORTAÇÃO PARA USO DE ESTADO

function Pesquisa(){

    const [textoDigitado, setTextoDigitado]= useState('');

    return(

        <PesquisaContainer>

            <Titulo>Já sabe por onde começar?</Titulo>

            <SubTitulo>Encontre seu livro em nossa estante.</SubTitulo>

            <Input

                placeholder="..."

                onBlur={evento => setTextoDigitado(evento.target.value)}

            />

            <p>{textoDigitado}</p>

        </PesquisaContainer>

    )

}

Tornando esse exemplo algo mais relacionado ao projeto, a ideia agora é mostrar na tela todos os livros cujo nome contenha o que for pesquisado no campo de pesquisa. Isso é feito primeiramente criando um arquivo dadosPesquisa.js, na mesma pasta do componente Pesquisa, e nesse arquivo declarar uma lista de objetos livros com atributos nome e id (para simular um banco de dados). Esse aquivo é importado para o index.js do componente Pesquisa, e nesse arquivo, as mudanças são feitas para adaptar o uso da lista.

import { livros } from './dadosPesquisa';//IMPORTAÇÃO DA LISTA DE OBJETOS livros

function Pesquisa(){

    const [livrosPesquisados, setLivrosPesquisados]= useState([]);

    var textoDigitado = '';

console.log(livrosPesquisados);

    return(

        <PesquisaContainer>

            <Titulo>Já sabe por onde começar?</Titulo>

            <SubTitulo>Encontre seu livro em nossa estante.</SubTitulo>

            <Input

                placeholder="..."

                onBlur={evento => {

                    textoDigitado = evento.target.value;

                    const resultadoPesquisa = livros.filter( livro =>

                        livro.nome.includes(textoDigitado));

                    setLivrosPesquisados(resultadoPesquisa);

                }}

            />

            <p>{textoDigitado}</p>

        </PesquisaContainer>

    )

}