React - CSS e Tailwind CSS

Link Notion: https://cherry-client-b8f.notion.site/React-CSS-e-Tailwind-CSS-885eac960ed44cbe8d64d62e2e6c6291?pvs=73

Dentro de projetos React, nós podemos utilizar diferentes abordagens para eatruturar o CSS.

CSS Global

Assim como em projetos sem utilização de bibliotecas e frameworks, nós podemos ter um arquivo CSS separado com todos os estilos, classes e o que mais for necessário e importá-lo no **main.jsx**.

E também podemos criar arquivos CSS separados, importando nos componentes correspodentes. Porém desta forma, apesar de ser importado no componente, **as regras do CSS permanecem globais, ou seja, não respeitam o escopo de componente**.

CSS Inline

No React, o CSS inline funciona de forma um pouco diferente do HTML tradicional, porque o style recebe um objeto JavaScript em vez de uma string.

```
import './App.css'
function App() {

  return (
    <>
    <h1
      style={
        {
          color: 'red',
          padding: '2rem'
        }
      }> React app</h1>
    <p> Este é um site em React.</p>
    </>
  )
}

export default App
```

Ele sempre terá duas chaves:

- style={{ color: 'red', padding: 2rem}} →
  - a primeira inicia o javascript dentro do HTML
  - a segunda é o objeto JS.

Por conta de ser um objeto Javascript, alguns padrões mudam em relação ao CSS.

- As propriedades CSS são escritas em camelCase (ex: backgroundColor em vez de background-color).
- Os valores numéricos que não têm unidade padrão (como px) precisam ser informados como string ou número. Se for número, o React assume px automaticamente (exceto em algumas propriedades específicas, como lineHeight).

CSS Modules

É uma técnica em que cada arquivo CSS é tratado como escopo local ao componente.

Ou seja, quando você importa um .module.css, o React gera identificadores únicos para as classes, evitando conflitos.

Isso significa que:

- As classes não vazam para o global.
- Você pode usar nomes simples (.container, .button) sem medo de sobrescrever outros componentes.

Como utilizar?

1. Crie um arquivo terminando com .module.css, por exemplo:

```
/* Button.module.css */
.button {
  background: green;
  color: white;
  padding: 10px 20px;
  border: none;
  border-radius: 6px;
  cursor: pointer;
}

.button:hover {
  background: darkgreen;
}
```

2. Importe no componente:

```
import styles from './Button.module.css';

function Button() {
  return <button className={styles.button}>Clique aqui</button>;
}

export default Button;
```

Você acessa cada classe como uma propriedade do objeto styles.

Ex.: .button vira **styles.button**.

Se a classe tiver hífen (.main-container), você acessa como string:

```
<div className={styles["main-container"]}></div>
```

3. No DOM, o React transforma a classe em algo único:

```
<button class="Button_button__a1b2c">Clique aqui</button>
```

Para combinar múltiplas classes, você deve concatená-las.

```
import styles from './Card.module.css';

function Card() {
  return (
    <div className={`${styles.card} ${styles.shadow}`}>
      Conteúdo do card
    </div>
  );
}
```

Styled Components

É uma biblioteca de CSS-in-JS, ou seja, você escreve o CSS diretamente dentro de arquivos JavaScript, criando componentes estilizados.

Em vez de criar um .css separado, você define estilos como partes do seu código React.

```jsx
import styled from "styled-components";

const Button = styled.button`
  background: green;
  color: white;
  padding: 10px 20px;
  border: none;
  border-radius: 6px;
  cursor: pointer;

  &:hover {
    background: darkgreen;
  }
`;

function App() {
  return <Button>Clique aqui</Button>;
}
```

Quando esse código roda, o Styled Components cria automaticamente classes únicas no DOM, garantindo que os estilos não vazem para outros componentes.

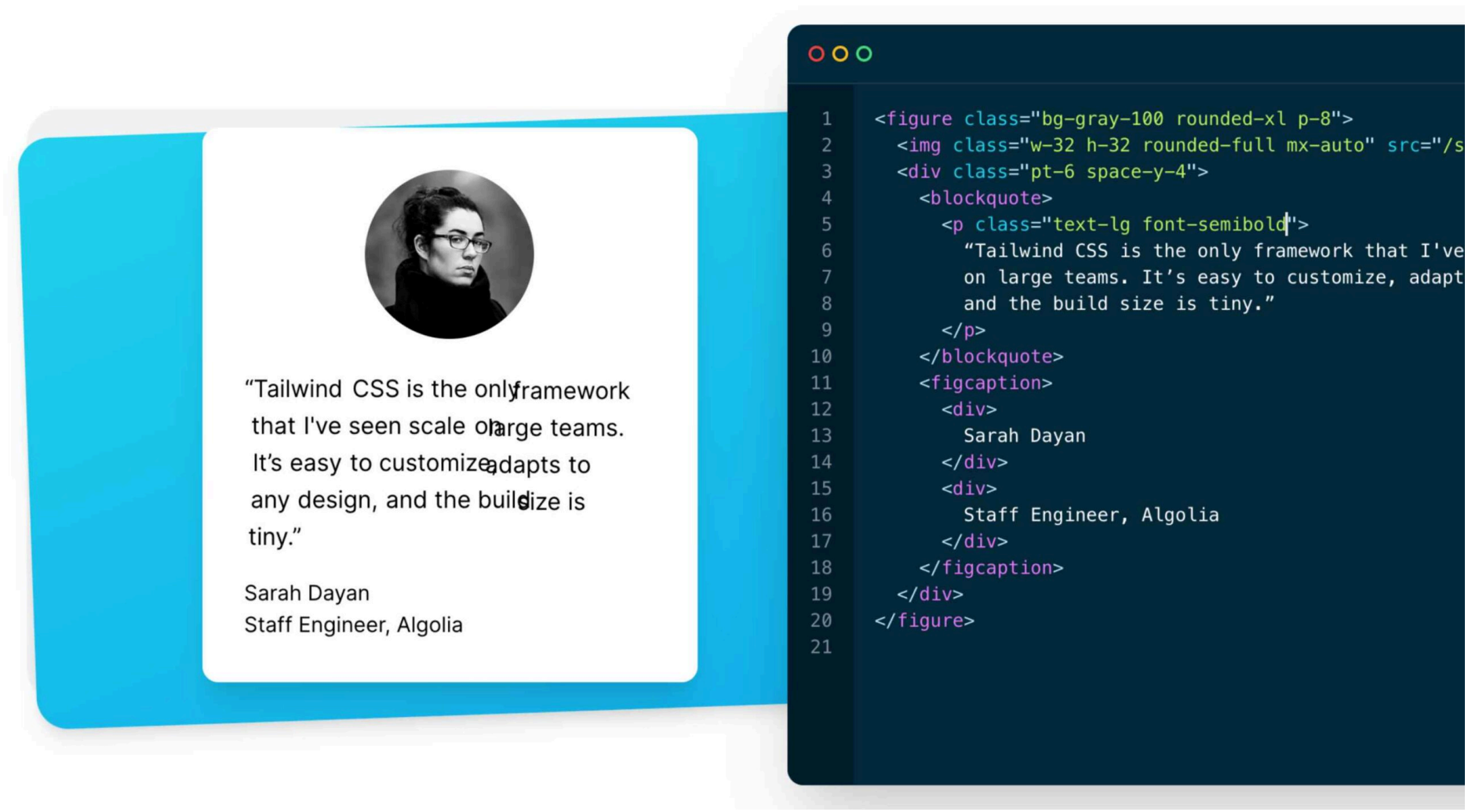
Para saber mais, acesse a documentação oficial:

https://styled-components.com/

Tailwind CSS

Tailwind é um framework CSS utilitário. Em vez de você escrever classes personalizadas no CSS, ele fornece centenas de classes prontas, cada uma representando uma regra de estilo pequena (cor, espaçamento, flex, grid, etc).

Você monta o estilo de um componente combinando essas classes direto no JSX.



```
function Card() {
  return (
    <div className="bg-white shadow-md rounded-lg p-6 max-w-sm">
      <h2 className="text-xl font-bold mb-2 text-gray-800">Título</h2>
      <p className="text-gray-600">Conteúdo do card.</p>
      <button className="mt-4 bg-blue-500 text-white px-4 py-2 rounded hover:bg-blue-600">
        Clique
      </button>
    </div>
  );
}
```

Neste exemplo:

- **bg-white** → fundo branco
- **shadow-md** → sombra média
- **rounded-lg** → bordas arredondadas grandes
- **p-6** → padding 1.5rem
- **mt-4** → margin-top 1rem
- **hover:bg-blue-600** → cor no hover

Instalando o Tailwind com o Vite

```
npm install tailwindcss @tailwindcss/vite
```

Altere o conteúdo do arquivo *tailwind.config.js,* adicionando o import do tailwind e a sua chamada dentro de plugins, deixando o conteúdo como abaixo:

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'
import tailwindcss from '@tailwindcss/vite'

// https://vite.dev/config/
export default defineConfig({
  plugins: [react(), tailwindcss()],
})
```

Remova tudo que tiver no index.css, deixando apenas:

```
@import "tailwindcss";
```

Com isso o Tailwind já deve estar funcionando no projeto.

Instalando o Tailwind CSS Intellisense

https://marketplace.visualstudio.com/items?itemName=bradlc.vscode-tailwindcss

O TailwindCSS Intellisense é a extensão para o VSCode que permite o autocomplete no código das classes do Tailwind, facilitando e adicionando produtividade no dia a dia.

Comandos básicos

text- texto

bg- background

m - margin (ml, mr, mt, mb, my, mx)

p - padding (pl, pr, pt, pb, py, px)

h - height

w - width

pseudo-classes - hover: , focus:

breakpoints (mobile-first) - sm, md, lg ,xl

flex/grid - flex, justify-between, grid-cols-2

Valores personalizados

Caso você precise aplicar um valor personalizado para uma propriedade, como uma cor ou tamanho específicos, você pode utilizar **os colchetes []**, para definir este valor.

```
<!-- Espaçamentos -->
<div class="p-[72px] m-[10%]">Caixa</div>

<!-- Tamanhos -->
<div class="w-[450px] h-[50vh] bg-blue-200">Custom</div>

<!-- Cores -->
<div class="bg-[#1DA1F2] text-[rgb(255,0,0)]">Cor customizada</div>

<!-- Bordas -->
<div class="border-[3px] border-[#ff00ff]">Borda</div>

<!-- Fontes -->
<p class="text-[32px] leading-[3.2rem] tracking-[0.2em]">
  Texto customizado
</p>
```

Adicionando propriedades personalizadas

O Tailwind permite uma alta personalização de suas classes e propriedades. Caso você precise adicionar cores, fontes e outras propriedades personalizadas, você deverá adicionar diretamente no arquivo CSS, dentro da diretiva @theme.

```
@import "tailwindcss";

@theme {
  --color-brand: #1DA1F2;
 }
```

Sendo composto por um prefixo (você pode saber mais sobre os prefixos na página oficial - https://tailwindcss.com/docs/theme) e o nome que você deseja para a propriedade. Neste caso, para utilizar a propriedade, bastaria utilizar a propriedade e o valor criado, por exemplo, utilizar a classe **bg-brand.**

Adicionando classes personalizadas

De acordo com a necessidade, seu código pode começar a ficar muito poluído quando exige muitas variações como estilos base, hover, media queries, dentro de um único elemento. Para isso, é possível criar classes personalizadas, reutilizando os estilos sempre que necessário.

O Tailwind organiza os estilos em **3 layers**:

- **base** → resets e estilos globais
- **components**→ padrões de UI
- **utilities** → classes utilitárias e overrides finais

```
@import "tailwindcss";

@theme {
   --color-brand: #1DA1F2;
 }

@layer base {
  h1 {
    @apply text-2xl;
  }
  h2 {
    @apply text-xl;
  }
}

@layer components {
  .btn-blue {
    @apply bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded;
  }
}

@layer utilities {
  .filter-none {
    filter: none;
  }
  .filter-grayscale {
    filter: grayscale(100%);
  }
```

Bibliotecas

https://www.framer.com/motion/

https://ui.shadcn.com/

https://headlessui.com/

Referências

https://www.freecodecamp.org/portuguese/news/o-que-e-tailwind-css-um-guia-para-iniciantes/

https://www.material-tailwind.com/

https://tailwindcss.com/

https://play.tailwindcss.com/