Dicas e truques styled-components

Durante o curso da Labenu, aprendemos sobre como estilizar componentes React usando a biblioteca *styled-components*. Essa biblioteca é amplamente utilizada no mercado por conversar bem com a lógica de componentes do React, possibilitando a estilização diretamente no Javascript.

Esse artigo busca trazer algumas dicas e truques sobre a biblioteca, trazendo formas de implementar funcionalidades conhecidas do CSS e que não são triviais com a biblioteca.

Pseudo-seletores (hover, active, etc)

No CSS, é possível usar pseudo-seletores para selecionar estados específicos de elementos na tela. O exemplo mais comum é o `hover`, que se refere ao elemento quando o usúario passa o mouse por cima dele. No CSS comum, os pseudo-seletores são usados da seguinte maneira:

```
div:hover { /* Estilos quando o usuário passa o mouse em cima */ }
```

Essa funcionalidade pode ser facilmente replicada com styled-components com a seguinte sintaxe:

```
import styled from 'styled-components' const ComponenteEstilizadoComHover
= styled.div` /* Outros estilos do componente */ :hover { /* Estilos
quando o usuário passa o mouse em cima */ } `
```

O mesmo pode ser aplicado para qualquer outro pseudo-seletor, como `active` ou `visited`, apesar de serem menos comuns.

Selecionando filhos de um elemento

No CSS comum, é possível combinar seletores selecionar elementos filhos de um seletor. Por exemplo, para selecionar todas as `div` filhas de um elemento com a classe `elemento`, poderíamos fazer:

```
.elemento div { /* Estilos para as divs dentro da classe elemento */ }
```

No caso do styled-components, como não trabalhamos diretamente com seletores, essa sintaxe não é possível. No entanto, é possível implementar isso, e a lógica é bastante semelhante com a dos pseudo-seletores mostrada acima.

```
import styled from 'styled-components' const ComponentePai = styled.div`
/* Outros estilos do componente */ div { /* Estilos para as divs dentro
do componente ComponentePai */ } `
```

Controlando estilos com Javascript

O styled-components, por estar dentro do Javascript, permite um controle de estilos direto com lógica de programação, algo que não era possível com o CSS puro.

Como o styled-components funciona por meio de componentes React, podemos passar informações para esses componentes por meio de **props**, e usar essas informações para definir a estilização.

Isso possibilita a criação de componentes mais reaproveitáveis e a reutilização da lógica do CSS.

Segue um exemplo de como passar informações para dentro do styled-component por meio de props:

```
import styled from 'styled-components' const ParagrafoEstilizado =
styled.p` color: ${(props) => props.cor}; ` const Componente = () => {
return <div> <ParagrafoEstilizado cor="blue">Meu texto
azul</ParagrafoEstilizado> <ParagrafoEstilizado cor="red">Meu texto
vermelho</ParagrafoEstilizado> </div> }
```

Também é possível implementar lógicas mais complexas dentro do componente estilizado:

```
import styled from 'styled-components' // Esse componente recebe uma prop
"lado". // Se ela for "direita", alinha os elementos a direita. // Se ela
for "esquerda", alinha os elementos a esquerda. // Caso contrário, alinha
os elementos no centro const FlexContainer = styled.div` display: flex;
${(props) => { if(props.lado === 'direita') { return 'justify-content:
flex-end;' } else if(props.lado === 'esquerda') { return 'justify-
content: flex-start;' } else { return 'justify-content: center;' } }} `
const Componente = () => { return <FlexContainer lado="direita"> <p>Texto
legal</p> </FlexContainer> }
```

Estilos globais

É comum iniciar o CSS com alguns estilos globais, que afetam a página inteira. Isso pode ser particularmente útil para resetar alguns estilos padrão de navegadores, como margens e paddings não desejados.

Como não temos arquivos CSS com styled-components, isso pode ser feito com uma função específica dele.

No arquivo App.js:

```
import { createGlobalStyle } from 'styled-components'; // Estilos
definidos aqui serão aplicados a toda a aplicação const GlobalStyle =
createGlobalStyle` body { margin: 0; padding: 0; } p { margin: 0; } /*
Outros estilos globais */ `; const App = () => { return <div> {/* Deve-se
colocar o componente GlobalStyle como primeiro componente do App */}
<GlobalStyle/> {/* Resto da aplicação */} </div> } export default App
```