eFront - styled-components

Introdução

A grande vantagem de se utilizar React como tecnologia de desenvolvimento é a sua componentização, isso nos ajuda a reaproveitar um pedaço de lógica em vários lugares da nossa aplicação. Com styled-components isso não é diferente, essa biblioteca funciona da mesma forma só que com estilos e utilizando o conceito CSS-in-JS. Além disso, a biblioteca styled-components permite que você escreva CSS simples para seus componentes sem se preocupar com colisões dos nomes de classes, assim, cada componente conhece apenas o seu CSS, uma mudança em um componente será refletida somente nele.

O que é styled-components?

Basicamente, styled components é uma biblioteca que utiliza o conceito de CSS-in-JS, ou seja, que nos permite escrever códigos CSS dentro do JavaScript ao nível de componente na sua aplicação. Com ele, você pode criar websites bonitos e funcionais. Além disso, ganhar mais agilidade e precisão no desenvolvimento web.

Um pouco mais sobre CSS-inJS: o método de estilização *CSS-in-JS* aplica estilos CSS em componentes individualmente em lugar de aplicá-los no documento HTML como um todo. Isso significa que a estilização CSS se aplica exclusivamente no escopo do componente e não no documento globalmente.

Vantagens de utilizar

Styled Components foi criado pelos seguintes motivos:

- **Automatização de CSS crítico**: styled-components controlam automaticamente quais componentes e seus respectivos estilos serão renderizados em uma página.
- **Eliminação de bugs devido a colisão de nomes de classes**: styled-components geram nomes de classe exclusivos para seus estilos. Você não precisa se preocupar com duplicação, sobreposição ou erros de ortografia.
- **Segurança na exclusão de CSS**: pode ser difícil saber se um nome de classe é usado em algum lugar na sua base de código. styled-components tornam isso óbvio, pois cada estilo está ligado a um componente específico. Se o componente não for utilizado ou for excluído, toda sua estilização será excluída.
- **Simplificação da estilização dinâmica**: adaptar o estilo de um componente com base em props ou em um tema global é uma tarefa simples, intuitiva e não requer gerenciamento manual de dezenas de classes.
- **Facilidade de manutenção**: você nunca precisará procurar em diferentes arquivos para encontrar o estilo que se aplica ao seu componente; portanto, a manutenção é muito fácil, não importa o tamanho da sua base de código.

Instalando

Já com o projeto React criado, iremos agora instalar o styled-components, para isso requer apenas um único comando para utilizar:

Com npm:
`npm install --save styled-components`

Com yarn:
`yarn add styled-components`

A documentação oficial do styled-components recomenda se você usa um gerenciador de pacotes como o yarn que oferece suporte ao campo "resolutions" no package.json. Isso ajuda a evitar problemas que surgem da instalação de várias versões do sytled-component em seu projeto.

No package.json:

```
{
 "resolutions": {
        "styled-components": "^5"
 }
}
```

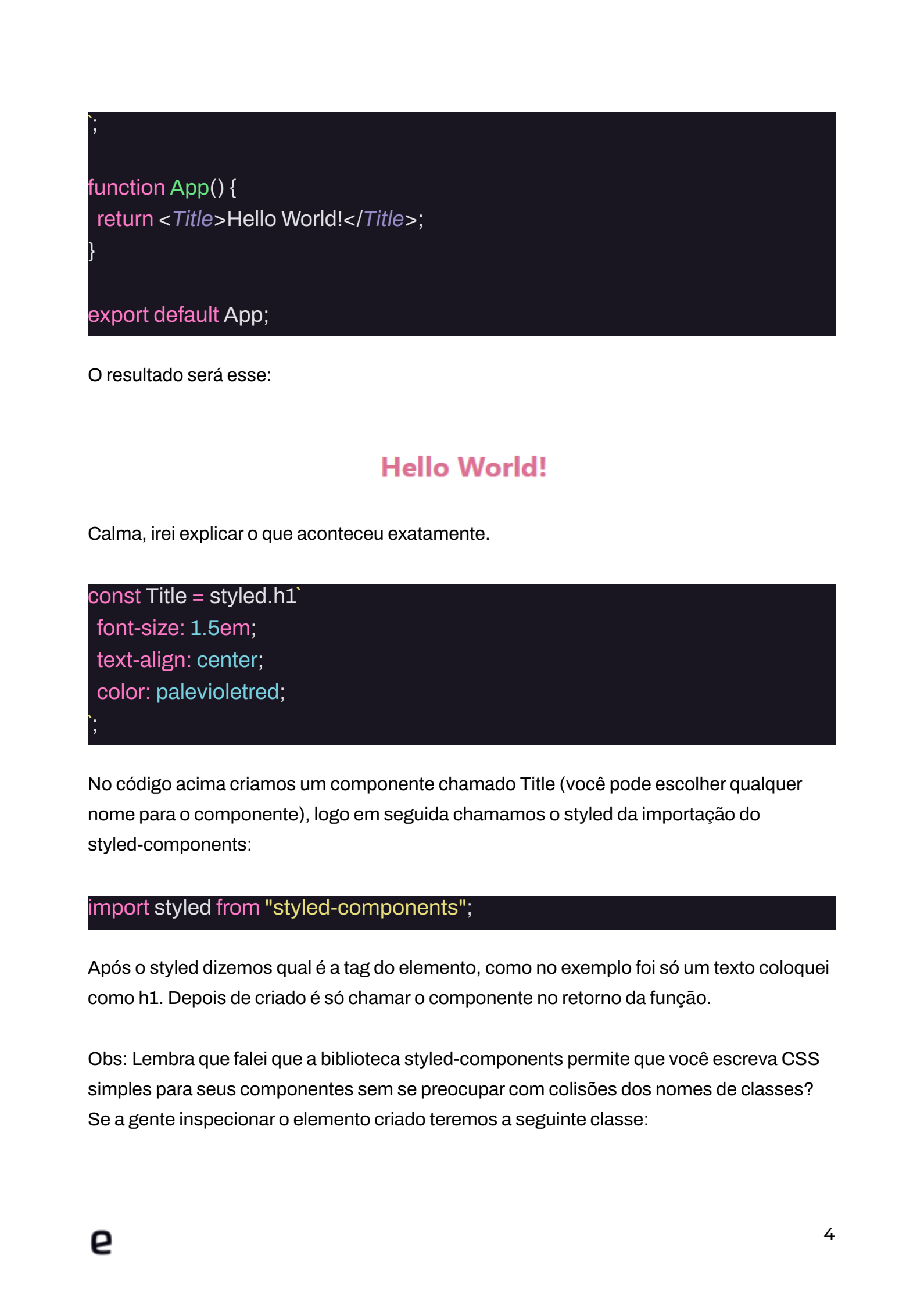
Aplicando Estilos

Depois de instalar a biblioteca vamos criar nosso primeiro componente. No arquivo App.js dentro da pasta src iremos ter o seguinte código:

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

const Title = styled.h1`
 font-size: 1.5em;
 text-align: center;
 color: palevioletred;
```

```
`;

function App() {
  return <Title>Hello World!</Title>;
}

export default App;
```

O resultado será esse:



Calma, irei explicar o que aconteceu exatamente.

```
const Title = styled.h1`
  font-size: 1.5em;
  text-align: center;
  color: palevioletred;
`;
```

No código acima criamos um componente chamado Title (você pode escolher qualquer nome para o componente), logo em seguida chamamos o styled da importação do styled-components:

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

Após o styled dizemos qual é a tag do elemento, como no exemplo foi só um texto coloquei como h1. Depois de criado é só chamar o componente no retorno da função.

Obs: Lembra que falei que a biblioteca styled-components permite que você escreva CSS simples para seus componentes sem se preocupar com colisões dos nomes de classes? Se a gente inspecionar o elemento criado teremos a seguinte classe:

Estilos dinâmicos

O styled-components nos permite passar uma função como o valor do estilo, isto é, passar estilos baseado nas props, ao invés de adicionar classes ao nosso arquivo, como é feito tradicionalmente com CSS.

Iremos utilizar o mesmo exemplo e mudar a cor do texto de forma dinâmica. Teremos o seguinte código:

```
const Title = styled.h1`
 font-size: 1.5em;
 text-align: center;
 color: ${(props) => (props.iuricode ? "green" : "palevioletred")};
`;
```

No exemplo acima (na propriedade color) fizemos uma verificação na props do componente Title. Caso tenha a propriedade “iuricode” iremos adicionar o cor verde (green) no elemento, caso não tenha iremos manter a mesma cor utilizada no primeiro exemplo.

Para um melhor entendimento irei renderizar dois componentes Title, dessa forma:

```
function App() {
 return (
  <div>
   <Title iuricode>Hello World!</Title>
   <Title>Hello World!</Title>
  </div>
```

```
  );
}
```

Como no primeiro componente chamado temos a prop “iuricode” teremos a seguinte renderização em nossa aplicação:

Estendendo estilos

O styled-components também permite estender estilos entre os componentes de forma bem simples.

```
const Title = styled.h1`
  font-size: 1.5em;
  text-align: center;
  color: palevioletred;
`;

const Title2 = styled(Title)`
  font-size: 4.5em;
`;
```

Dessa forma acabamos de criar um componente chamado “Title2” e adicionamos os mesmos estilos do componente “Title”, porém aumentamos o tamanho da fonte. O resultado é esse:

Criando estilos em arquivos separados

“Okay Iuri, mas é estranho colocar os estilos no mesmo arquivo da lógica da aplicação!” É, de fato é mesmo! Para resolver isso, o styled-components nos dá a possibilidade de separar a lógica do estilo, isso é, criando arquivos separados.

Iremos criar um arquivo chamado “Title.js”, nele iremos adicionar todos os estilos já utilizados nos exemplos anteriores. Porém, antes de “const” na criação do componente iremos colocar o “export” para assim exportar nosso arquivo de estilo no arquivo de lógica.

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

export const Title = styled.h1`
 font-size: 1.5em;
 text-align: center;
 color: palevioletred;
`;
```

Feito isso basta agora importar dessa forma no nosso arquivo App.js

```
import { Title } from "./Title.js";
```

Okay Iuri, mas toda vez que eu tiver um componente/elemento vou ter que importar ele assim? Se todos os elementos tiver dentro de um elemento pai não será preciso criar

componente por componente no styled-components, pois ele funciona da mesma forma do Sass, basta chamar o elemento pai.

No exemplo a seguir temos uma tag span dentro do componente Title:

```
<Title>
    Hello World! <span>texto maior</span>
</Title>
```

Para estilizar a tag span bata chamar ele dentro do componente Tiltle, dessa forma:

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

export const Title = styled.h1`
 font-size: 1.5em;
 text-align: center;
 color: palevioletred;

 span {
  font-size: 4.5em;
 }
`;
```

Teremos o seguinte resultado porém sem precisar criar um componente para o span:

Aplicando estilos globais

Em muitos casos queremos aplicar estilos globais em nossa aplicação. Dentro do da pasta src crie um arquivo chamado globalStyles.js

Nele, usaremos uma função chamado createGlobalStyle do styled-components e adicionaremos alguns estilos globais, dessa forma:

```
import { createGlobalStyle } from "styled-components";

export const GlobalStyle = createGlobalStyle`
 body {
  margin: 0;
  padding: 0;
  font-family: Open-Sans, Helvetica, Sans-Serif;
  background-color: #121212;
 }
`;
```

Agora iremos importar o componente GlobalStyle no index.js para que seja aplicado em todos documentos.

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import { GlobalStyle } from "./globalStyles.js";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
 <React.StrictMode>
  <GlobalStyle />
  <App />
 </React.StrictMode>
);
```

Dessa forma todo estilo adicionado no arquivo globalStyles.js será aplicado de forma global em nossa aplicação.

Resultado: