



FRONT-END DESIGN ENGINEERING



[aula 03]

[Atualizado em: 23/01/2024]



[React – Estilização – Styled Components]



REACT – ESTILIZAÇÃO – STYLED COMPONENTS

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

- ✓ **Styled Components** é uma biblioteca para React e React Native que permite escrever estilos CSS diretamente dentro dos seus componentes JavaScript, utilizando uma sintaxe similar à de literais de template.
- ✓ Em vez de definir estilos em arquivos CSS separados ou utilizar outras soluções como CSS-in-JS, Styled Components permite escrever seus estilos junto com a lógica dos seus componentes.

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

- ✓ Aqui estão alguns conceitos-chave sobre Styled Components:
- ✓ **Componentes Estilizados:** Em vez de definir estilos globais, Styled Components permite definir estilos específicos para componentes individuais. Você pode criar um componente estilizado usando a função `styled` fornecida pelo Styled Components. Por exemplo, você pode criar um botão estilizado assim:

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

```
import styled from 'styled-components';
```

```
const Button = styled.button`
```

```
  background-color: #007bff;
```

```
  color: #fff;
```

```
  border: none;
```

```
  border-radius: 4px;
```

```
  padding: 10px 20px;
```

```
  cursor: pointer;
```

```
  &:hover {
```

```
    background-color: #0056b3;
```

```
  }
```

```
`;
```

```
// Uso do componente Button
```

```
<Button>Clique Aqui</Button>
```

É um acento 'crase'

É um acento 'crase'

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

- ✓ Aqui estão alguns conceitos-chave sobre Styled Components:
- ✓ **Interpolação de Propriedades:** Você pode passar props para seus componentes estilizados e utilizar essas props dentro dos seus estilos. Isso permite uma maior personalização e reutilização de componentes. Por exemplo, você pode alterar a cor de fundo de um botão com base em uma prop primary:

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

```
const Button = styled.button`
  background-color: ${props => props.primary ? '#007bff' : '#fff'};
  color: ${props => props.primary ? '#fff' : '#007bff'};
  border: 1px solid #007bff;
  border-radius: 4px;
  padding: 10px 20px;
  cursor: pointer;
  &:hover {
    background-color: ${props => props.primary ? '#0056b3' : '#f0f0f0'};
  }
`;

// Uso do componente Button
<Button primary>Botão Primário</Button>
<Button>Botão Secundário</Button>
```

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

- ✓ Aqui estão alguns conceitos-chave sobre Styled Components:
- ✓ **Estilos Globais:** Embora Styled Components incentive o uso de estilos a nível de componente, ele também oferece uma maneira de definir estilos globais usando o componente `createGlobalStyle`. Isso permite aplicar estilos a todo o seu documento.

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

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

```
const GlobalStyle = createGlobalStyle`
```

```
  body {
```

```
    font-family: Arial, sans-serif;
```

```
    background-color: #f0f0f0;
```

```
  }
```

```
`;
```

```
// Para usar na sua aplicação
```

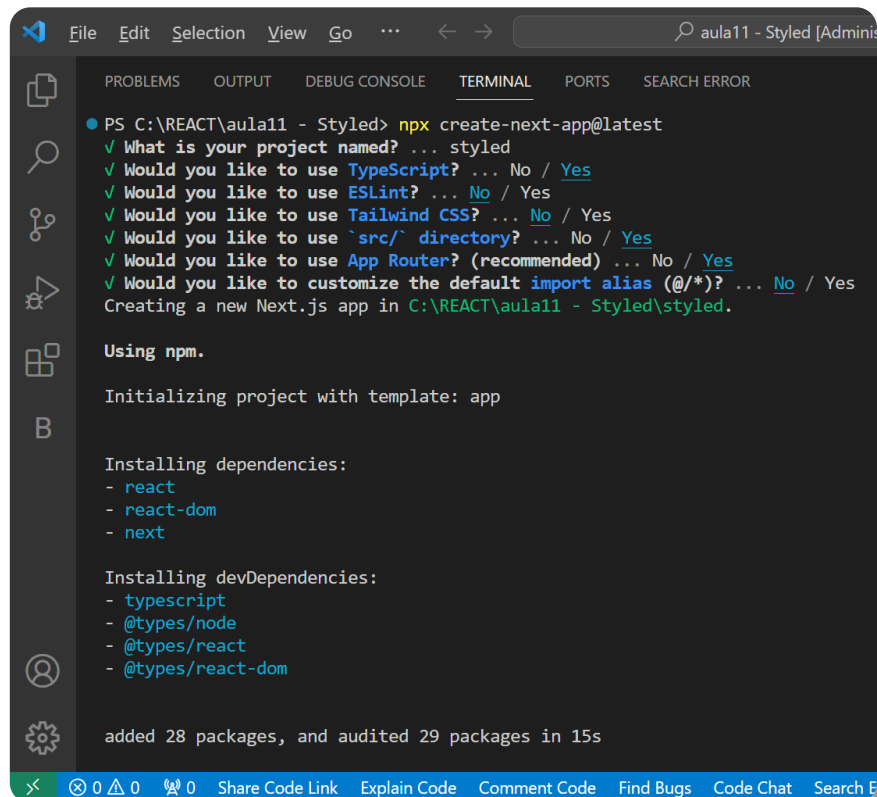
```
<GlobalStyle />
```

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

- ✓ Em resumo, **Styled Components** simplifica a gestão de estilos em suas aplicações React, permitindo que você defina estilos juntamente com seus componentes, facilitando a criação de interfaces de usuário coerentes e de fácil manutenção.

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

✓ Vamos criar nossa aplicação



```
PS C:\REACT\aula11 - Styled> npx create-next-app@latest
✓ What is your project named? ... styled
✓ Would you like to use TypeScript? ... No / Yes
✓ Would you like to use ESLint? ... No / Yes
✓ Would you like to use Tailwind CSS? ... No / Yes
✓ Would you like to use `src/` directory? ... No / Yes
✓ Would you like to use App Router? (recommended) ... No / Yes
✓ Would you like to customize the default import alias (@/*)? ... No / Yes
Creating a new Next.js app in C:\REACT\aula11 - Styled\styled.

Using npm.

Initializing project with template: app

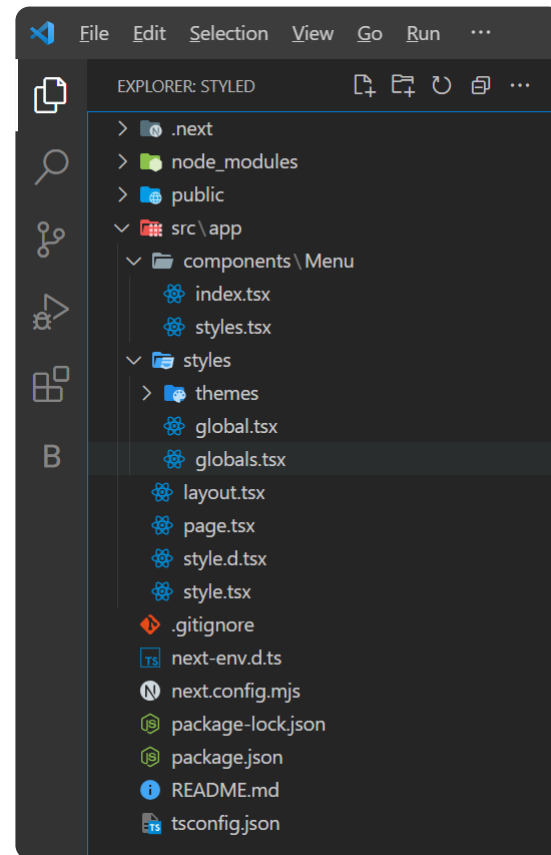
Installing dependencies:
- react
- react-dom
- next

Installing devDependencies:
- typescript
- @types/node
- @types/react
- @types/react-dom

added 28 packages, and audited 29 packages in 15s
```

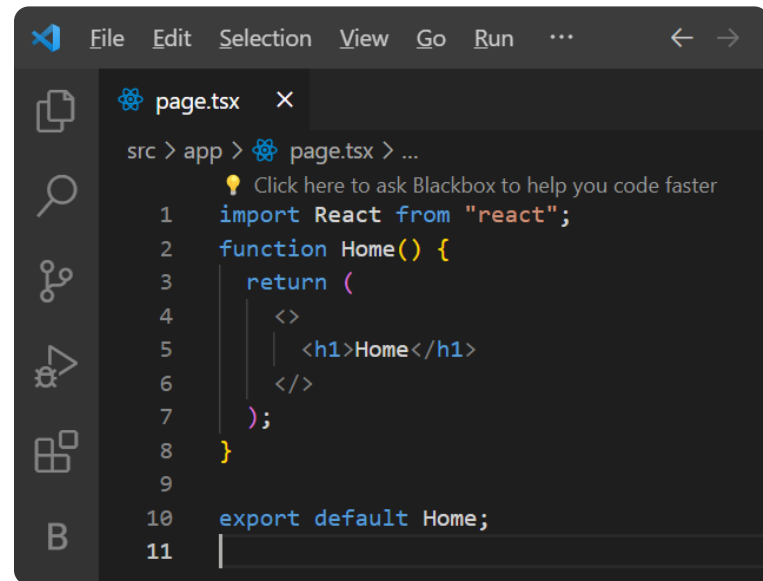
REACT – ESTILIZAÇÃO – STYLED COMPONENTS

- ✓ **Vamos criar nossa aplicação**
- ✓ **Nossa aplicação final terá a seguinte estrutura**



REACT – ESTILIZAÇÃO – STYLED COMPONENTS

- ✓ Vamos criar nossa aplicação
- ✓ Modificando o componente page.tsx



```
src > app > page.tsx > ...  
Click here to ask Blackbox to help you code faster  
1 import React from "react";  
2 function Home() {  
3   return (  
4     <>  
5       <h1>Home</h1>  
6     </>  
7   );  
8 }  
9  
10 export default Home;  
11
```

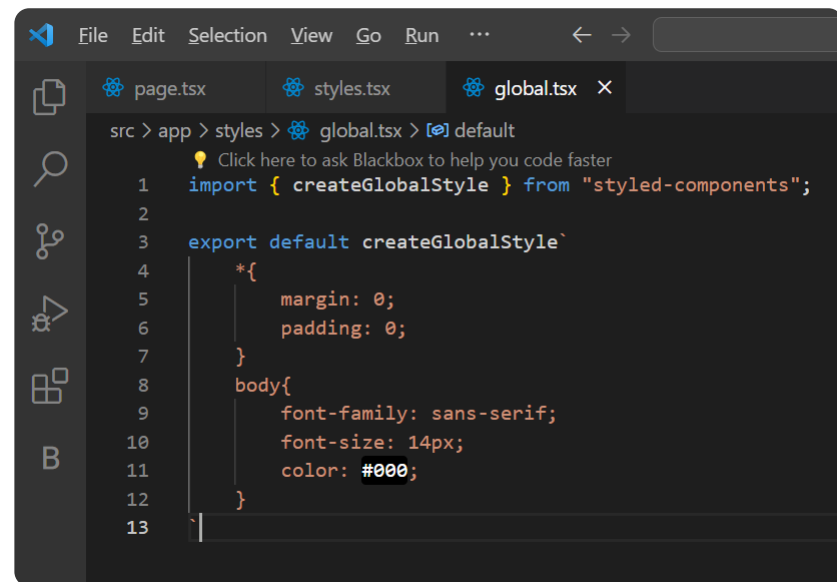
REACT – ESTILIZAÇÃO – STYLED COMPONENTS

- ✓ **Vamos criar nossa aplicação**
- ✓ Começaremos criando um estilo global para a nossa aplicação, para isso precisamos primeiro adicionar ao nosso projeto a biblioteca do Styled Components, e como dito antes ela ajuda muito na estilização dos componentes, e é com ele que vamos criar os temas da nossa página.
- ✓ Mas antes precisamos instalar o style-components

```
npm install styled-components
```

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

- ✓ **Vamos criar nossa aplicação**
- ✓ Depois de concluído podemos começar criando um arquivo de estilo global para a página, para isso criamos um diretório "styles" dentro da pasta "src", e dentro dela o arquivo "globals.tsx", e fica dessa forma.

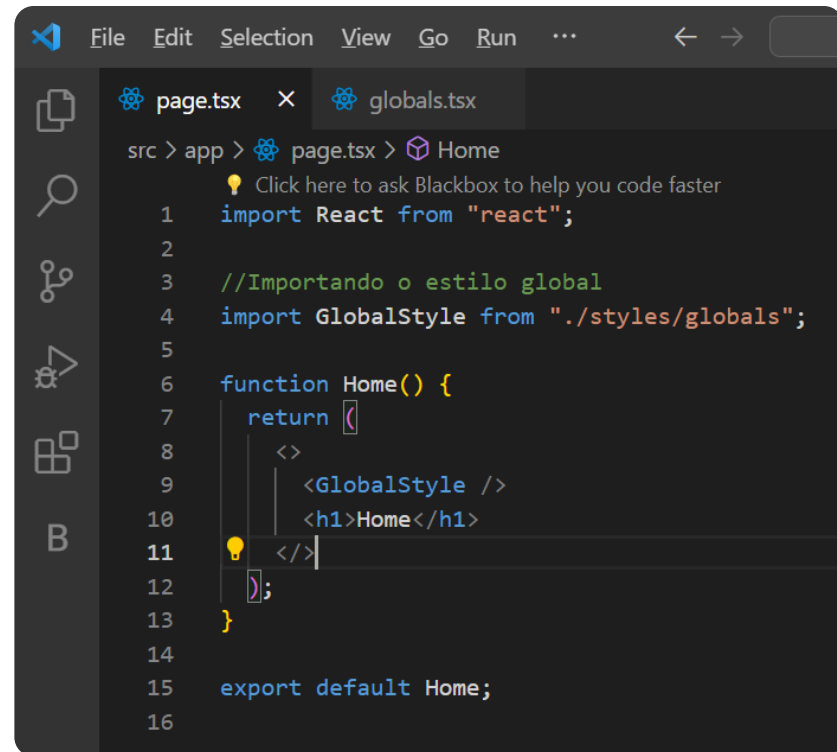


The screenshot shows a code editor with three tabs: page.tsx, styles.tsx, and global.tsx. The global.tsx tab is active, showing the following code:

```
src > app > styles > global.tsx > default
Click here to ask Blackbox to help you code faster
1 import { createGlobalStyle } from "styled-components";
2
3 export default createGlobalStyle`
4   *{
5     margin: 0;
6     padding: 0;
7   }
8   body{
9     font-family: sans-serif;
10    font-size: 14px;
11    color: #000;
12  }
13`
```

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

- ✓ **Vamos criar nossa aplicação**
- ✓ E agora podemos importar esse arquivo para dentro do nosso "Page.tsx" que as configurações de layout já devem estar funcionando.



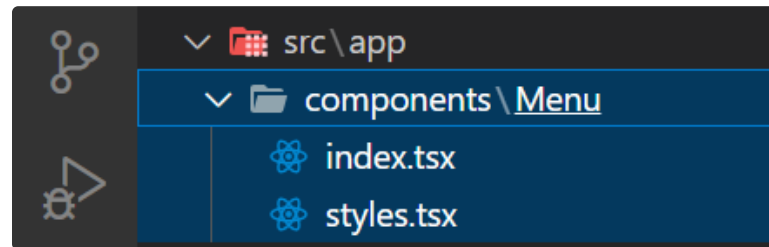
```
1 import React from "react";
2
3 //Importando o estilo global
4 import GlobalStyle from "../styles/globals";
5
6 function Home() {
7   return (
8     <>
9       <GlobalStyle />
10      <h1>Home</h1>
11    </>
12  );
13 }
14
15 export default Home;
```


REACT – ESTILIZAÇÃO – STYLED COMPONENTS

✓ Vamos criar nossa aplicação

✓ Pronto, ao fim disso tudo, podemos ver que a aplicação começou a ter uma cara um pouco mais agradável.

✓ Agora, vamos continuar criando um diretório "components" dentro da pasta "src/app" do projeto, é nesse diretório onde vamos colocar todos os nossos componentes que criarmos.

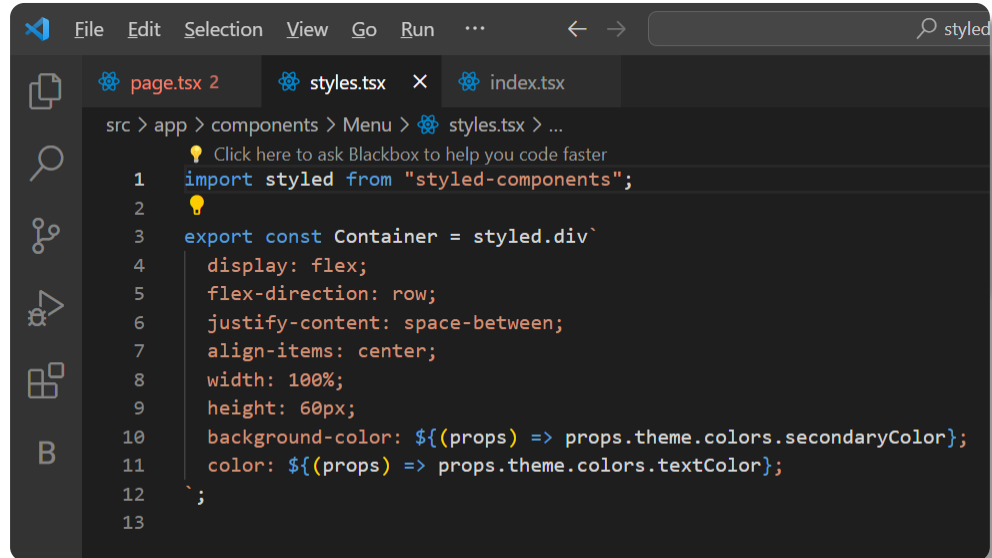


✓ Vamos começar criando o componente "Menu", essa organizar dessa forma os componentes:

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

✓ Vamos criar nossa aplicação

✓ O arquivo de estilização ficará assim:

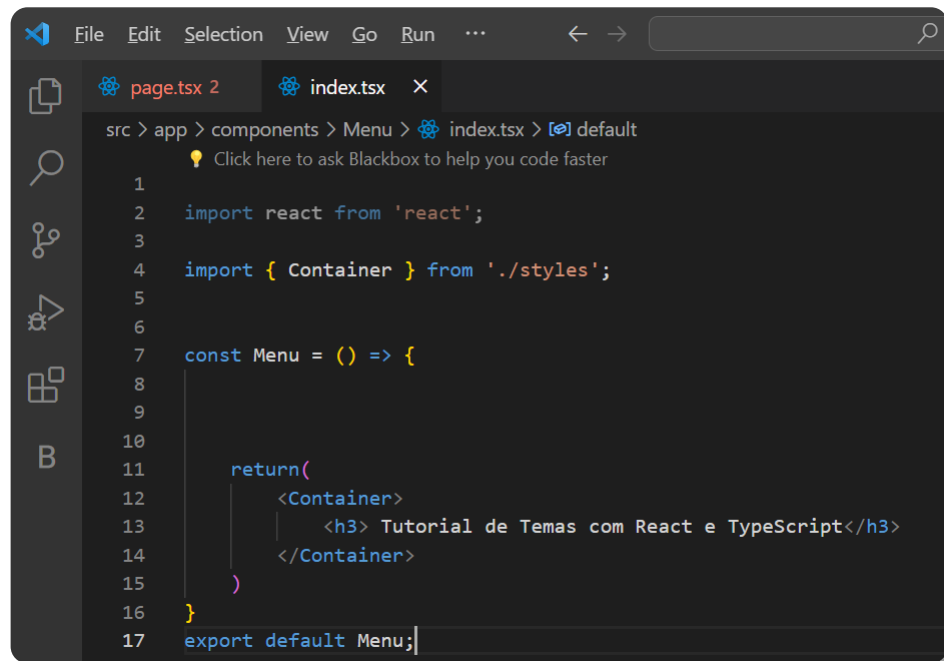


The screenshot shows a code editor with three tabs: page.tsx 2, styles.tsx, and index.tsx. The active tab is styles.tsx, which contains the following code:

```
src > app > components > Menu > styles.tsx > ...  
1 Click here to ask Blackbox to help you code faster  
1 import styled from "styled-components";  
2  
3 export const Container = styled.div`  
4   display: flex;  
5   flex-direction: row;  
6   justify-content: space-between;  
7   align-items: center;  
8   width: 100%;  
9   height: 60px;  
10  background-color: ${(props) => props.theme.colors.secondaryColor};  
11  color: ${(props) => props.theme.colors.textColor};  
12 `;  
13
```

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

- ✓ **Vamos criar nossa aplicação**
- ✓ O arquivo index.tsx ficará assim:

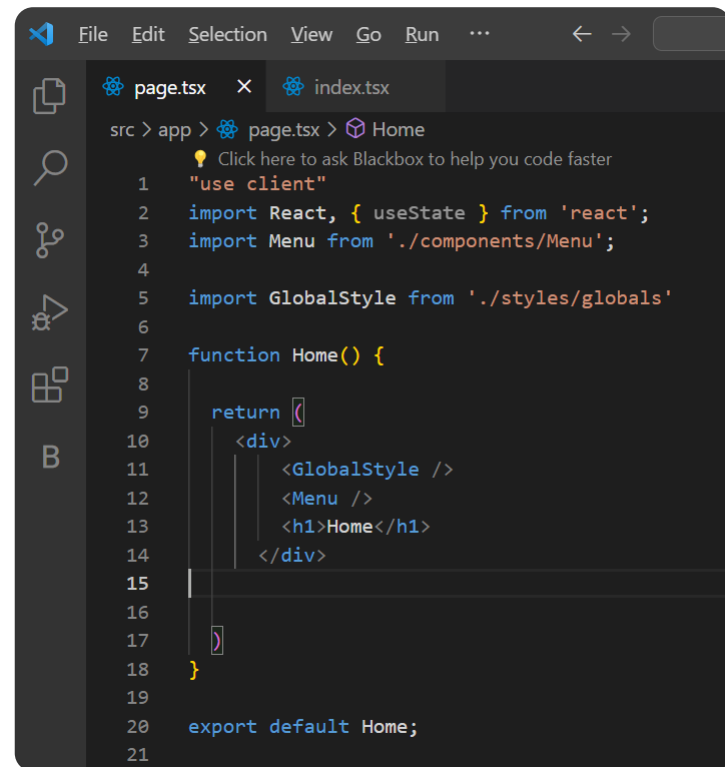


```
src > app > components > Menu > index.tsx > default
Click here to ask Blackbox to help you code faster
1
2 import react from 'react';
3
4 import { Container } from './styles';
5
6
7 const Menu = () => {
8
9
10
11   return(
12     <Container>
13       <h3> Tutorial de Temas com React e TypeScript</h3>
14     </Container>
15   )
16 }
17 export default Menu;
```

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

✓ Vamos criar nossa aplicação

- ✓ Depois de finalizado iremos importar esse componente para a página principal do nosso projeto, o "page.tsx"



```
File Edit Selection View Go Run ...  
page.tsx x index.tsx  
src > app > page.tsx > Home  
Click here to ask Blackbox to help you code faster  
1 "use client"  
2 import React, { useState } from 'react';  
3 import Menu from './components/Menu';  
4  
5 import GlobalStyle from './styles/globals'  
6  
7 function Home() {  
8  
9   return (  
10     <div>  
11       <GlobalStyle />  
12       <Menu />  
13       <h1>Home</h1>  
14     </div>  
15   )  
16  
17 }  
18  
19  
20 export default Home;  
21
```

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

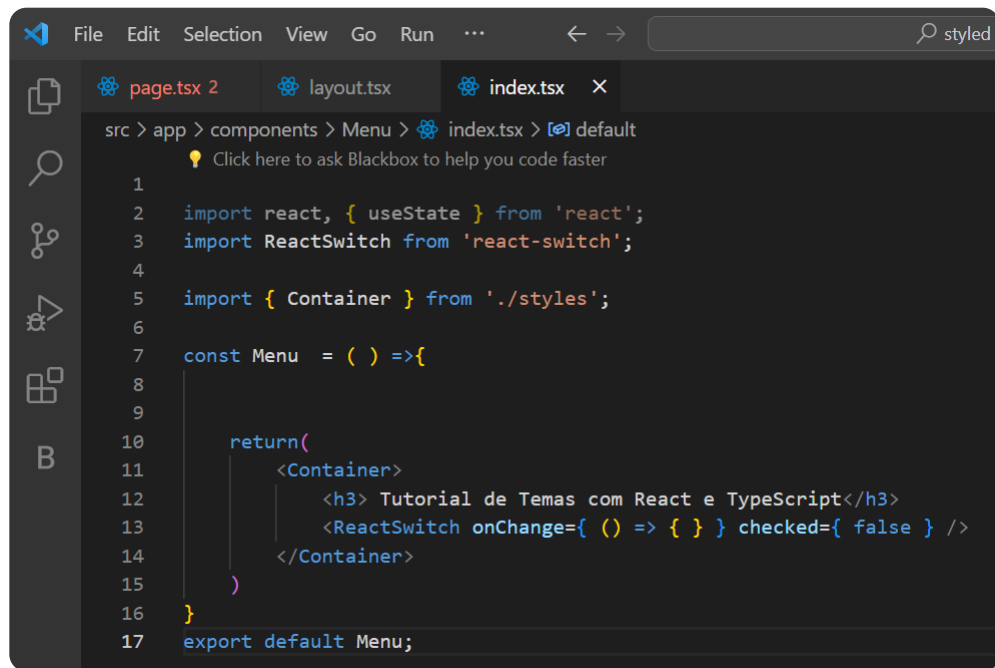
- ✓ **Vamos criar nossa aplicação**
- ✓ Para continuar, vamos instalar um carinha que vai fazer com que a gente consiga trocar de um tema para outro, o react-switch traz pra gente um componente pronto, e nos poupa do trabalho de criar um componente do zero.

```
npm install react-switch
```

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

✓ Vamos criar nossa aplicação

- ✓ Ao fim da instalação já podemos implementar na nossa página, como o foco desse conteúdo é tema dinâmico no React vamos nos aprofundar nesse componente, pois ele é apenas um suporte para agilizar o nosso processo de desenvolvimento.
- ✓ Dentro do nosso componente "Menu", podemos adicionar o ReactSwitch da seguinte forma:



```
File Edit Selection View Go Run ... ← → styled [
page.tsx 2 layout.tsx index.tsx X
src > app > components > Menu > index.tsx > default
Click here to ask Blackbox to help you code faster
1
2 import react, { useState } from 'react';
3 import ReactSwitch from 'react-switch';
4
5 import { Container } from './styles';
6
7 const Menu = ( ) =>{
8
9
10   return(
11     <Container>
12       <h3> Tutorial de Temas com React e TypeScript</h3>
13       <ReactSwitch onChange={ ( ) => { } } checked={ false } />
14     </Container>
15   )
16 }
17 export default Menu;
```

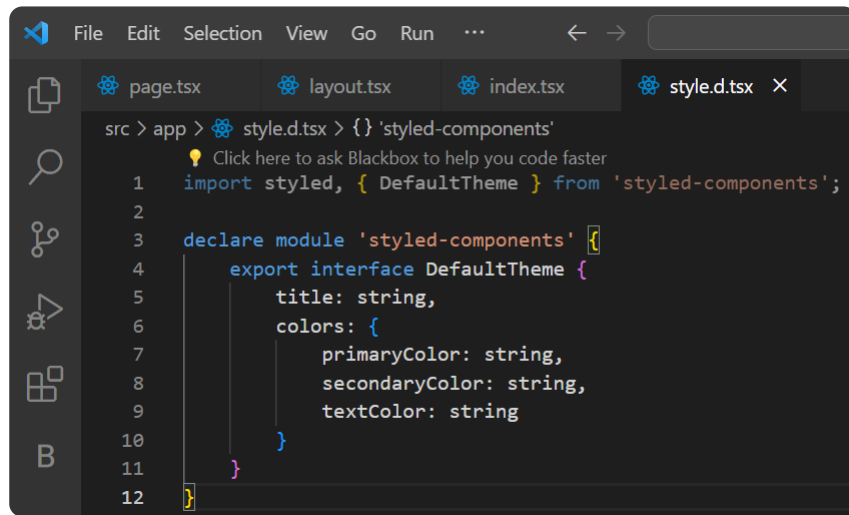
REACT – ESTILIZAÇÃO – STYLED COMPONENTS

✓ Vamos criar nossa aplicação

- ✓ Podemos notar que o nosso componente já foi adicionado, porém ainda não está funcional, pois para que ele funcione corretamente primeiro precisamos criar os nossos temas.
- ✓ Para começar a criar os temas precisamos criar um arquivo "styled.d.ts" para que a gente consiga "sobrescrever" uma interface do próprio Styled Components, você pode criar esse arquivo em qualquer nível de diretório do seu projeto, por organização optei por criar dentro do diretório src.
- ✓ Dentro dessa interface "DefaultTheme" vamos criar o modelo que os nossos temas devem seguir, como esse é um tutorial focado em como criar temas com Styled Components, não vamos nos aprofundar muito na parte de estilização, então nossos temas vão ser bem simples contendo apenas as cores de fundo e de textos.

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

- ✓ Vamos criar nossa aplicação

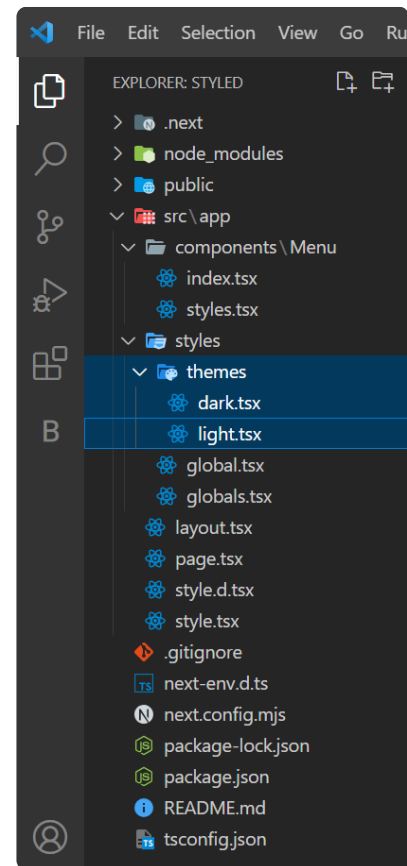


The screenshot shows a code editor with a dark theme. The top bar includes a menu (File, Edit, Selection, View, Go, Run, ...) and navigation arrows. The tab bar shows four files: page.tsx, layout.tsx, index.tsx, and style.d.tsx (selected). The editor content shows the following code:

```
src > app > style.d.tsx > {} 'styled-components'
  Click here to ask Blackbox to help you code faster
1  import styled, { DefaultTheme } from 'styled-components';
2
3  declare module 'styled-components' {
4    export interface DefaultTheme {
5      title: string,
6      colors: {
7        primaryColor: string,
8        secondaryColor: string,
9        textColor: string
10     }
11   }
12 }
```


REACT – ESTILIZAÇÃO – STYLED COMPONENTS

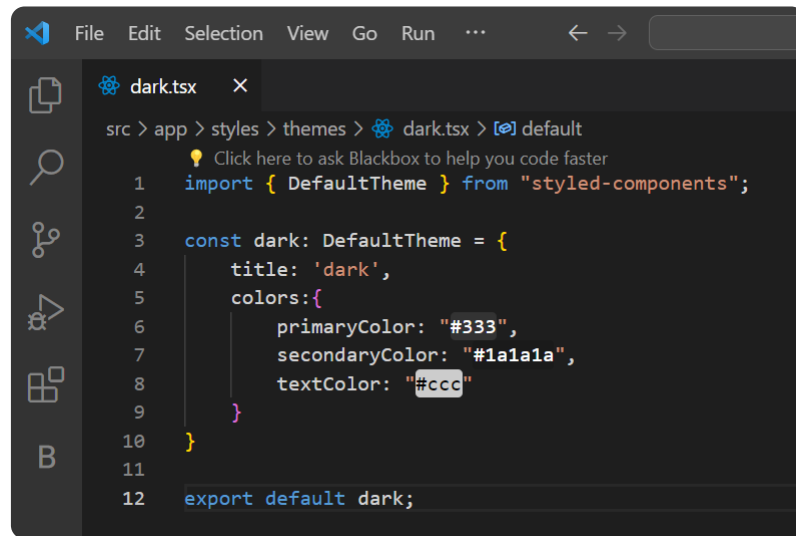
- ✓ **Vamos criar nossa aplicação**
- ✓ Depois de terminar de fazer o modelo dos seus temas, podemos então criar um diretório "themes" dentro do diretório "styles", dentro da pasta "themes" criaremos dois arquivos de temas, o tema dark e o light que são os temas que iremos utilizar aqui.



REACT – ESTILIZAÇÃO – STYLED COMPONENTS

✓ Vamos criar nossa aplicação

✓ Tema: dark.tsx



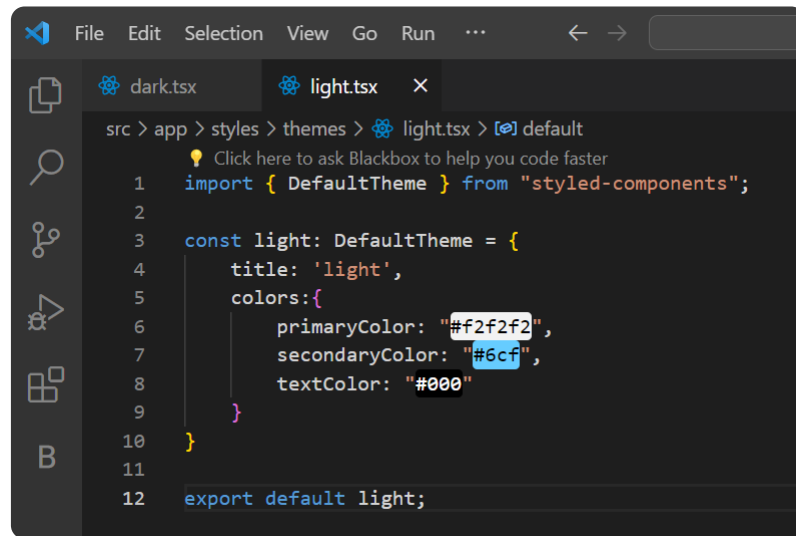
The screenshot shows a code editor with a dark theme. The file explorer on the left shows the project structure: src > app > styles > themes > dark.tsx. The main editor area displays the content of dark.tsx, which imports DefaultTheme from styled-components and defines a dark theme object with title, colors, primaryColor, secondaryColor, and textColor properties. The code is as follows:

```
1 import { DefaultTheme } from "styled-components";
2
3 const dark: DefaultTheme = {
4   title: 'dark',
5   colors: {
6     primaryColor: "#333",
7     secondaryColor: "#1a1a1a",
8     textColor: "#ccc"
9   }
10 }
11
12 export default dark;
```

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

✓ Vamos criar nossa aplicação

✓ Tema: light.tsx



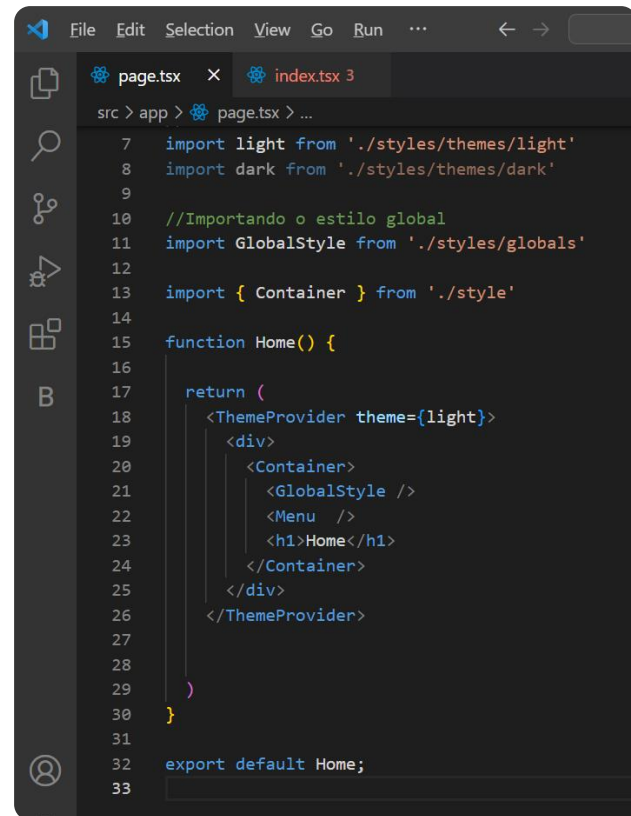
The screenshot shows a VS Code editor window with the file explorer on the left. The active file is `light.tsx` located at `src > app > styles > themes > light.tsx`. The code defines a light theme using `styled-components`.

```
1 import { DefaultTheme } from "styled-components";
2
3 const light: DefaultTheme = {
4   title: 'light',
5   colors: {
6     primaryColor: "#f2f2f2",
7     secondaryColor: "#6cf",
8     textColor: "#000"
9   }
10 }
11
12 export default light;
```

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

✓ Vamos criar nossa aplicação

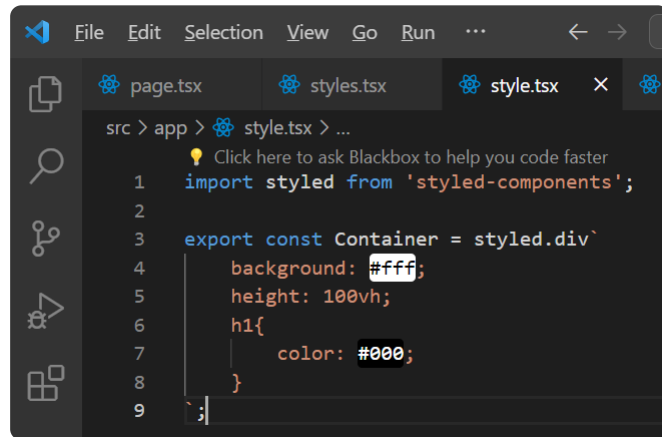
- ✓ Depois de finalizado, nossos temas já estão feitos, o que falta é apenas utilizá-los, para fazer isso precisamos voltar no nosso arquivo "page.tsx", pois tem algumas coisas que precisamos modificar para que o React consiga aplicar os temas corretamente.
- ✓ Utilizaremos agora um componente novo, o "ThemeProvider", é ele que faz os temas funcionarem no React, e para implementar ele é bem simples, precisamos apenas importar para o nosso arquivo e colocá-lo em volta de tudo que vai ser modificado quando o tema for trocado, ele tem uma propriedade chamada "theme" e é nessa propriedade que iremos passar o nosso tema desejado, então não esqueça de importar também os temas para o arquivo.



```
7 import light from './styles/themes/light'
8 import dark from './styles/themes/dark'
9
10 //Importando o estilo global
11 import GlobalStyle from './styles/globals'
12
13 import { Container } from './style'
14
15 function Home() {
16
17   return (
18     <ThemeProvider theme={light}>
19       <div>
20         <Container>
21           <GlobalStyle />
22           <Menu />
23           <h1>Home</h1>
24         </Container>
25       </div>
26     </ThemeProvider>
27   )
28
29 }
30
31 export default Home;
32
33
```

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

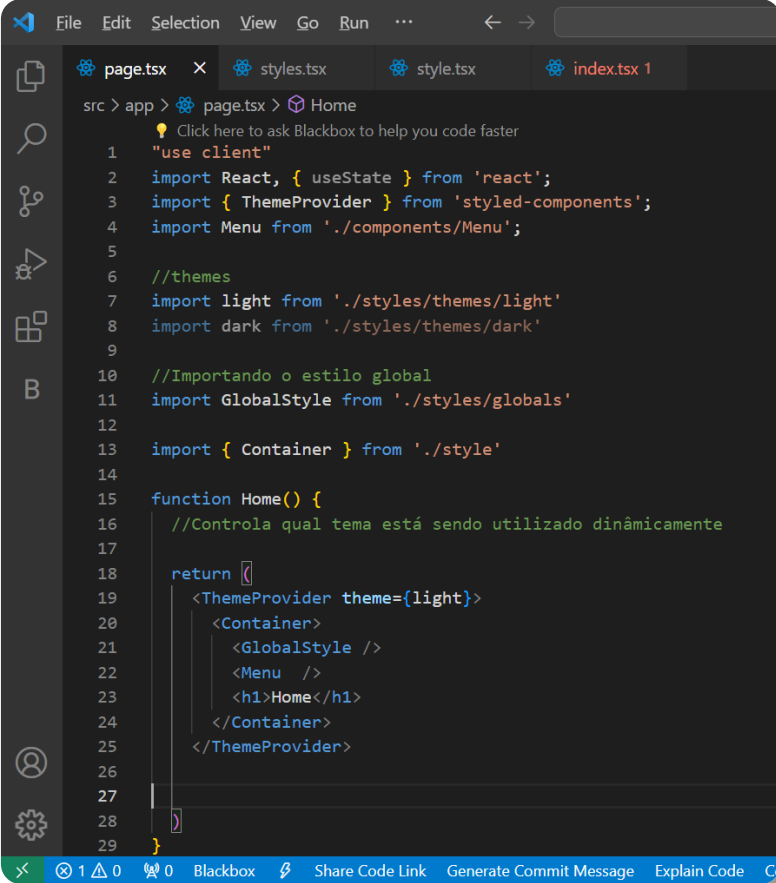
- ✓ **Vamos criar nossa aplicação**
- ✓ Definimos que o tema inicial seria o tema light, mas fique a vontade para trocar por outro de sua preferência, OK?
- ✓ Bom feito isso já está quase tudo pronto, porém ainda não conseguimos ver o tema funcionando na página.
- ✓ Bom isso acontece porque não estamos utilizando o tema dentro dos nossos styles, e depois que não criamos ainda um style para a nossa página principal.



```
src > app > style.tsx > ...  
Click here to ask Blackbox to help you code faster  
1 import styled from 'styled-components';  
2  
3 export const Container = styled.div`  
4   background: #fff;  
5   height: 100vh;  
6   h1{  
7     color: #000;  
8   }  
9 `;
```

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

- ✓ **Vamos criar nossa aplicação**
- ✓ Pronto, style da página criado com sucesso, agora é só importar esse arquivo para a página home (page.tsx) e substituir a "div" que tem nela por Container, que é o componente que acabamos de criar no arquivo acima. Dessa forma:

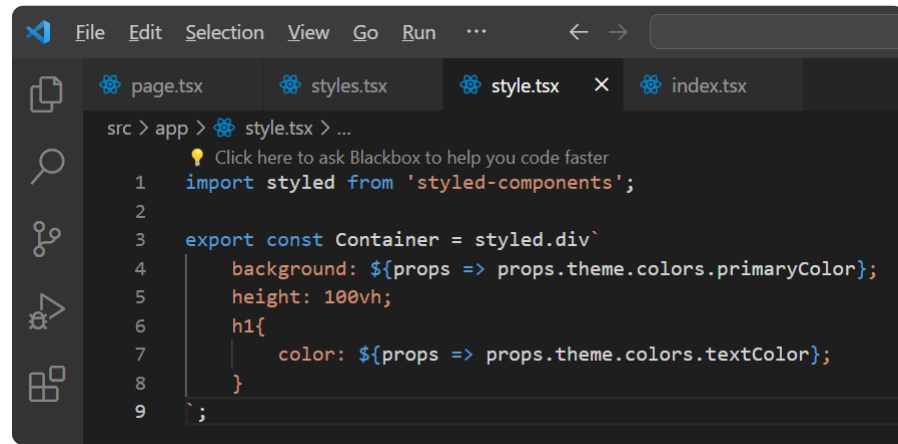


```
File Edit Selection View Go Run ...  
page.tsx x styles.tsx style.tsx index.tsx 1  
src > app > page.tsx > Home  
Click here to ask Blackbox to help you code faster  
1 "use client"  
2 import React, { useState } from 'react';  
3 import { ThemeProvider } from 'styled-components';  
4 import Menu from './components/Menu';  
5  
6 //themes  
7 import light from './styles/themes/light'  
8 import dark from './styles/themes/dark'  
9  
10 //Importando o estilo global  
11 import GlobalStyle from './styles/globals'  
12  
13 import { Container } from './style'  
14  
15 function Home() {  
16   //Controla qual tema está sendo utilizado dinamicamente  
17  
18   return (  
19     <ThemeProvider theme={light}>  
20       <Container>  
21         <GlobalStyle />  
22         <Menu />  
23         <h1>Home</h1>  
24       </Container>  
25     </ThemeProvider>  
26   )  
27  
28  
29 }
```

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

✓ Vamos criar nossa aplicação

- ✓ Pronto, depois de finalizado já podemos colocar nossos temas em ação, e é bem simples basta apenas você substituir o que está fixo no código por uma variável dentro dos arquivos de estilos.
- ✓ Os componentes feitos com styled components conseguem receber props como todos os outros, e é com essas props que conseguiremos deixar as cores do nosso site dinâmicas.
- ✓ Começando pelos styles da home, fica dessa forma:



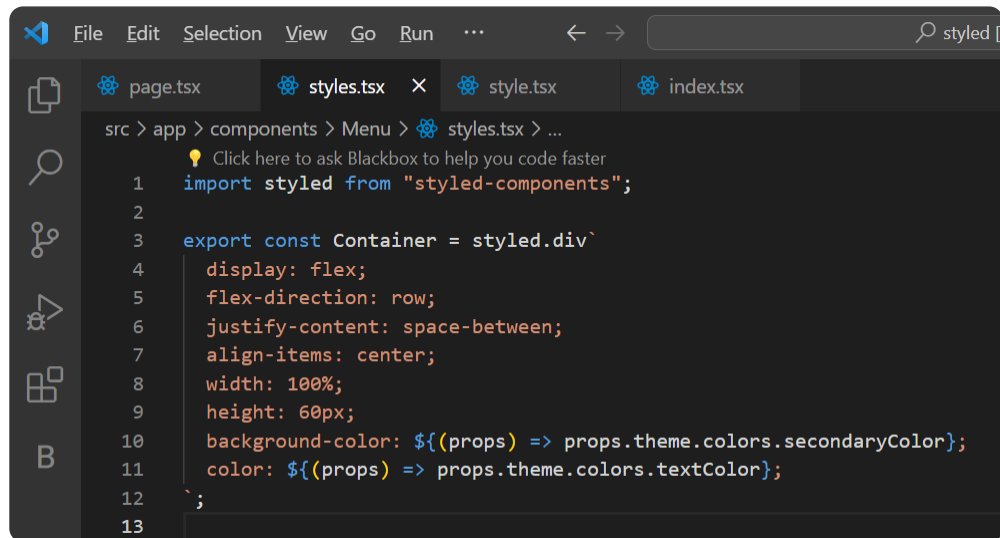
```
File Edit Selection View Go Run ... < >
page.tsx styles.tsx style.tsx index.tsx
src > app > style.tsx > ...
Click here to ask Blackbox to help you code faster
1 import styled from 'styled-components';
2
3 export const Container = styled.div`
4   background: ${props => props.theme.colors.primaryColor};
5   height: 100vh;
6   h1{
7     color: ${props => props.theme.colors.textColor};
8   }
9 `;
```

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

- ✓ **Vamos criar nossa aplicação**
- ✓ Dessa forma as cores que irão ficar no background e a cor do texto vai depender de qual arquivo você está enviando para o ThemeProvider, da forma que está agora, as coisas não devem mudar muito, pois as cores do tema light é basicamente as cores que estavam fixas antes.
- ✓ No arquivo de estilo do menu, adicionamos uma nova propriedade css para mudar a cor do texto, e também trocamos as cores fixas por variáveis vindas das props.

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

- ✓ Vamos criar nossa aplicação



The screenshot shows a VS Code editor window with a dark theme. The top bar includes the VS Code logo and menu items: File, Edit, Selection, View, Go, Run, and a search bar. Below the menu, there are four tabs: page.tsx, styles.tsx (active), style.tsx, and index.tsx. The main editor area shows the file path: src > app > components > Menu > styles.tsx > A tooltip提示 says "Click here to ask Blackbox to help you code faster". The code in the editor is as follows:

```
1 import styled from "styled-components";
2
3 export const Container = styled.div`
4   display: flex;
5   flex-direction: row;
6   justify-content: space-between;
7   align-items: center;
8   width: 100%;
9   height: 60px;
10  background-color: ${(props) => props.theme.colors.secondaryColor};
11  color: ${(props) => props.theme.colors.textColor};
12 `;
13
```

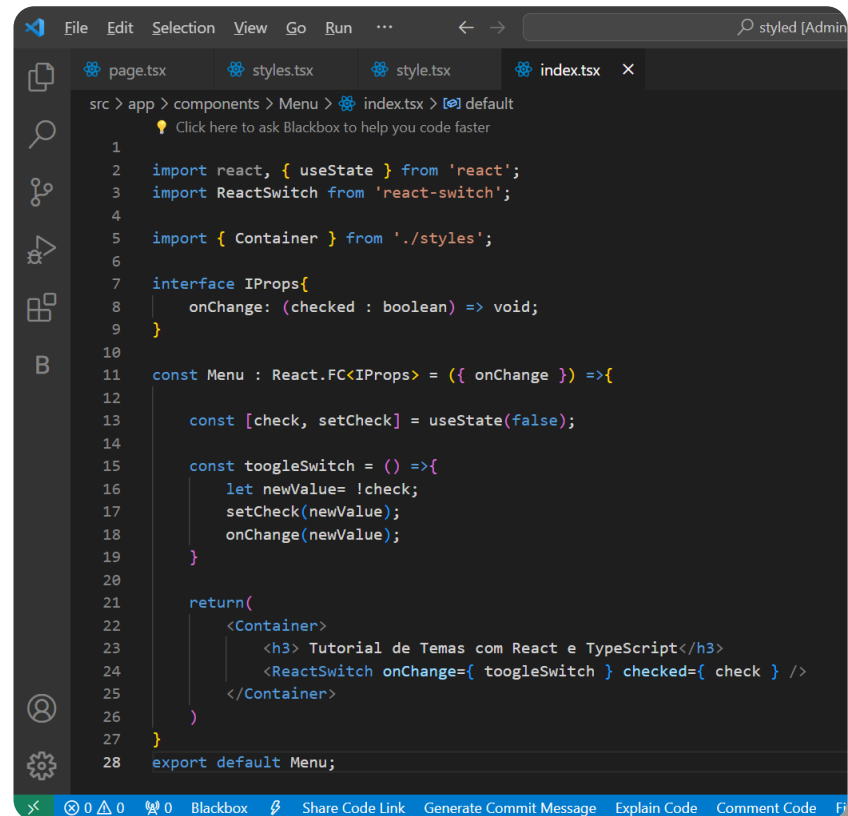
REACT – ESTILIZAÇÃO – STYLED COMPONENTS

✓ Vamos criar nossa aplicação

- ✓ Por fim, vamos colocar esse componente para modificar automaticamente quando o switch for pressionado, pois no momento é preciso que você modifique os arquivos manualmente.
- ✓ E no React sabemos que quando se trata de variáveis com valores dinâmicos, estamos falando de estados, então iremos utilizar o hook **useState** para fazer esse controle para nós, para isso precisamos modificar um pouquinho o arquivo do componente de menu, para que o switch consiga enviar a informação do tema pra tela de home, OK?

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

✓ Vamos criar nossa aplicação

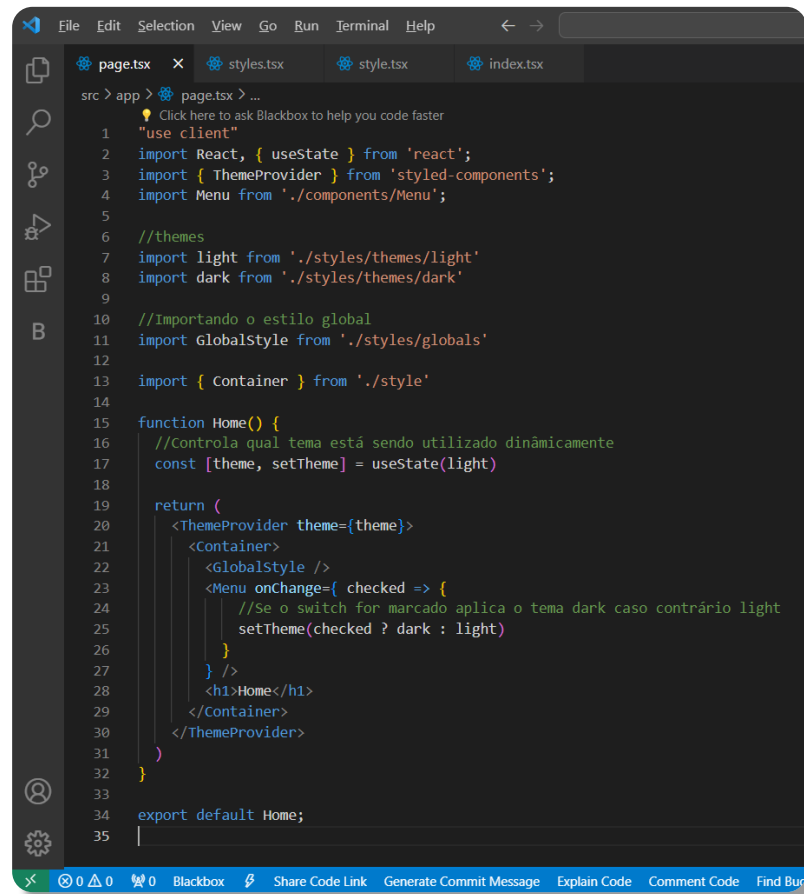


The screenshot shows a code editor with the following code in `index.tsx`:

```
1 import react, { useState } from 'react';
2 import ReactSwitch from 'react-switch';
3
4 import { Container } from './styles';
5
6
7 interface IProps{
8   onChange: (checked : boolean) => void;
9 }
10
11 const Menu : React.FC<IProps> = ({ onChange }) =>{
12
13   const [check, setCheck] = useState(false);
14
15   const toggleSwitch = () =>{
16     let newValue= !check;
17     setCheck(newValue);
18     onChange(newValue);
19   }
20
21   return(
22     <Container>
23       <h3> Tutorial de Temas com React e TypeScript</h3>
24       <ReactSwitch onChange={ toggleSwitch } checked={ check } />
25     </Container>
26   )
27 }
28 export default Menu;
```

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

- ✓ **Vamos criar nossa aplicação**
- ✓ **Modificamos também o page.tsx**



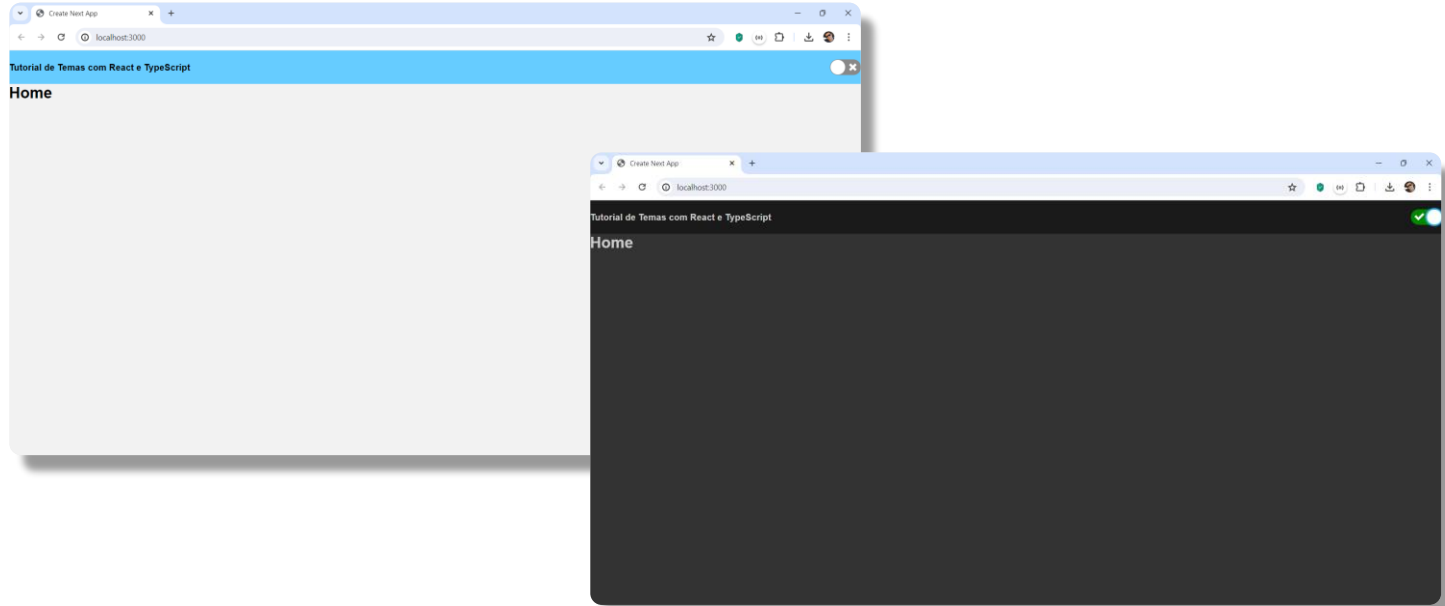
```
1  Click here to ask Blackbox to help you code faster
2  "use client"
3  import React, { useState } from 'react';
4  import { ThemeProvider } from 'styled-components';
5  import Menu from './components/Menu';
6
7  //themes
8  import light from './styles/themes/light'
9  import dark from './styles/themes/dark'
10
11 //Importando o estilo global
12 import GlobalStyle from './styles/globals'
13
14 import { Container } from './style'
15
16 function Home() {
17   //Controla qual tema está sendo utilizado dinamicamente
18   const [theme, setTheme] = useState(light)
19
20   return (
21     <ThemeProvider theme={theme}>
22       <Container>
23         <GlobalStyle />
24         <Menu onChange={ checked => {
25           //Se o switch for marcado aplica o tema dark caso contrário light
26           setTheme(checked ? dark : light)
27         }} />
28         <h1>Home</h1>
29       </Container>
30     </ThemeProvider>
31   )
32 }
33
34 export default Home;
35
```

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

- ✓ **Vamos criar nossa aplicação**
- ✓ E finalmente terminamos nosso pequeno projeto de temas com o React, você já consegue perceber que ao clicar no switch o tema da página é trocado dinamicamente.

REACT – ESTILIZAÇÃO – STYLED COMPONENTS

- ✓ **Vamos criar nossa segunda aplicação REACT**
- ✓ E no seu navegador deverá ser carregada a seguinte página.



REACT – EXERCÍCIOS

✓ Exercício 01 - Estilização em React: Construindo um Card Interativo

- ✓ Você foi designado para criar um componente React que represente um card interativo. Este card deve exibir informações fictícias sobre um produto e ter interações visuais quando o usuário passar o mouse sobre ele.
- ✓ Requisitos:
 - ✓ Crie um componente funcional React chamado InteractiveCard.
 - ✓ O card deve exibir as seguintes informações fictícias:
 - ✓ Nome do produto: "Smartphone XYZ"
 - ✓ Preço: R\$ 999,99
 - ✓ Descrição: "O smartphone XYZ é repleto de recursos incríveis para atender às suas necessidades diárias."
 - ✓ Estilize o card de forma agradável e responsiva usando CSS-in-JS.
 - ✓ Adicione uma animação de transição suave para as interações do mouse. Quando o usuário passar o mouse sobre o card, as cores de fundo e texto devem mudar gradualmente.
 - ✓ Certifique-se de que o card seja visualmente atraente e que as informações sejam legíveis.
 - ✓ Dicas:
 - ✓ Utilize as propriedades de estilo do React para aplicar estilos diretamente ao componente.
 - ✓ Considere a utilização de propriedades de estado para controlar as interações do mouse.
 - ✓ Teste diferentes combinações de cores, sombras e tamanhos para criar um design atraente.

REFERÊNCIAS BIBLIOGRÁFICAS

ANTONIO, C. Pro React: Build Complex Front-End Applications in a Composable Way With React. Apress, 2015.

BOSWELL, D; FOUCHER, T. The Art of Readable Code: Simple and Practical Techniques for Writing Better Code. Estados Unidos: O'Reilly Media, 2012.

BRITO, Robin Cris. Android Com Android Studio - Passo A Passo. Editora Ciência Moderna.

BUNA, S. React Succinctly. Estados Unidos: [s.n], 2016. Disponível em: <www.syncfusion.com/ebooks/reactjs_succinctly>. Acesso em: 12 de janeiro de 2023.

FACEBOOK (2019a). React: Getting Started. React Docs, 2019. Disponível em: <reactjs.org/docs/react-api.html>. Acesso em: 13 de janeiro de 2023.

FACEBOOK (2019b). React Without ES6. React Docs, 2019. Disponível em: <reactjs.org/docs/react-without-es6.html>. Acesso em: 10 de janeiro de 2023.

FACEBOOK (2019c). React Without JSX. React Docs, 2019. Disponível em: <reactjs.org/docs/react-without-jsx.html>. Acesso em: 10 de janeiro de 2023.

FREEMAN, Eric ROBSON, Elisabeth. Use a Cabeça! Programação em HTML5. Rio de Janeiro: Editora Alta Books, 2014

GACKENHEIMER, C. Introduction to React: Using React to Build scalable and efficient user interfaces.[s.i.]: Apress, 2015.

HUDSON, P. Hacking with React. 2016. Disponível em: <www.hackingwithreact.com/read/1/3/introduction-to-jsx>. Acesso em: 13 janeiro de 2023.

KOSTRZEWA, D. Is React.js the Best JavaScript Framework in 2018? 2018. Disponível em: <hackernoon.com/is-react-js-the-best-javascript-framework-in-2018-264a0eb373c8>. Acesso em: janeiro de 2023.

MARTIN, R. Clean Code: A Handbook of Agile Software Craftsmanship. Estados Unidos: Prentice Hall, 2009.

MDN WEB DOCS. Guia JavaScript. Disponível em <<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide>>. Acessado em 29 de janeiro de 2023.

REFERÊNCIAS BIBLIOGRÁFICAS

NELSON, J. Learn React's Fundamentals Without the Buzzwords? 2018. Disponível em: <jamesknelson.com/learn-react-fundamentals-sans-buzzwords>. Acesso em: 12 janeiro de 2023.

NIELSEN, J. Response Times: The 3 Important Limits. 1993. Disponível em: <www.nngroup.com/articles/response-times-3-important-limits>. Acesso em: 10 janeiro de 2023.

O'REILLY, T. What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. 2005. Disponível em: <www.oreilly.com/pub/a/web2/archive/what-is-web-20.html#mememap>. Acesso em: 10 de janeiro de 2023.

PANDIT, N. What Is ReactJS and Why Should We Use It? 2018. Disponível em: <www.c-sharpcorner.com/article/what-and-why-reactjs>. Acesso em: 12 de janeiro de 2023.

RAUSCHMAYER, A. Speaking JavaScript: An In-Depth Guide for Programmers. Estados Unidos: O'Reilly Media, 2014.

REACTIVA. O arquivo package-lock.json. Disponível em: <<https://nodejs.reativa.dev/0020-package-lock-json/index>>. Acessado em 13 de janeiro de 2023.

_____. O guia do package.json. Disponível em: <<https://nodejs.reativa.dev/0019-package-json/index>>. Acessado em 13 de janeiro de 2023.

RICOY, L. Desmitificando React: Uma Reflexão para Iniciantes. 2018. Disponível em: <medium.com/trainingcenter/desmitificando-react-uma-reflex%C3%A3o-para-iniciantes-a57af90b6114>. Acesso em: 13 janeiro de 2023.

SILVA, Maurício Samy. Ajax com jQuery: requisições Ajax com a simplicidade de jQuery. São Paulo: Novatec Editora, 2009.

_____. Construindo Sites com CSS e XHTML. Sites Controlados por Folhas de Estilo em Cascata. São Paulo: Novatec, 2010.

_____. CSS3 - Desenvolva aplicações web profissionais com o uso dos poderosos recursos de estilização das CSS. São Paulo: Novatec Editora, 2010.

STACKOVERFLOW. Most Popular Technologies: Web Frameworks. Developer Survey Results, StackOverflow, 2019. Disponível em: <insights.stackoverflow.com/survey/2019#technology>. Acesso em: 13 de janeiro de 2023.

REFERÊNCIAS BIBLIOGRÁFICAS

W3C. HTML5 - A linguagem de marcação que revolucionou a web. São Paulo: Novatec Editora, 2010.

_____. A vocabulary and associated APIs for HTML and XHTML. Disponível em <<https://www.w3.org/TR/2018/SPSD-html5-20180327/>>. Acessado em 28 de abril de 2020, às 20h53min.

_____. Cascading Style Sheets, level 1. Disponível em <<https://www.w3.org/TR/2018/SPSD-CSS1-20180913/>>. Acessado em 28 de abril de 2020, às 21h58min.

_____. Cascading Style Sheets, level 2 Revision 2. Disponível em <<https://www.w3.org/TR/2016/WD-CSS22-20160412/>>. Acessado em 28 de abril de 2020, às 22h17min.

_____. Cascading Style Sheets, level 2. Disponível em <<https://www.w3.org/TR/2008/REC-CSS2-20080411/>>. Acessado em 28 de abril de 2020, às 22h03min.

_____. Cascading Style Sheets, level 3. Disponível em <<https://www.w3.org/TR/css-syntax-3/>>. Acessado em 28 de abril de 2020, às 22h18min.

_____. HTML 3.2 Reference Specification. Disponível em <<https://www.w3.org/TR/2018/SPSD-html32-20180315/>>. Acessado em 28 de abril de 2020, às 19h37min.

_____. HTML 4.0 Specification. Disponível em <<https://www.w3.org/TR/1998/REC-html40-19980424/>>. Acessado em 28 de abril de 2020, às 19h53min.

_____. HTML 4.01 Specification. Disponível em <<https://www.w3.org/TR/2018/SPSD-html401-20180327/>>. Acessado em 28 de abril de 2020, às 20h04min.

_____. Cascading Style Sheets, level 2 Revision 1. Disponível em <<https://www.w3.org/TR/CSS2/>>. Acessado em 28 de abril de 2020, às 22h13min.

WIKIPEDIA. JavaScript. Disponível em <<https://pt.wikipedia.org/wiki/JavaScript>>. Acessado em 29 de abril de 2020, às 10h.

- Dúvidas?
- Críticas?
 - Sugestões?
 - Ameaças?