

Componentes Funcionais e styled-components

Labenu_



O que vamos ver hoje?

- O que é um componente?
- Criando um componente
- Usando componentes dentro de outros
- Props
- styled-components



Conceito

Labenu_



Relembrando - Funções

- Bloco de código reutilizável
- Podemos dar **nomes** e **significados** à partes específicas do código

Entrada
(Input)



Saída
(Output)



Componentes

- Componentes são blocos de código reutilizáveis
- Agora esses blocos representam um pedaço de **interface**, “gerando” o HTML que aparece na página

Entrada
(**Props**)



Saída
(**JSX**)



[Início](#)[Em alta](#)[Inscrições](#)[Originais](#)[Biblioteca](#)[Histórico](#)

Titulo do video



Titulo do video



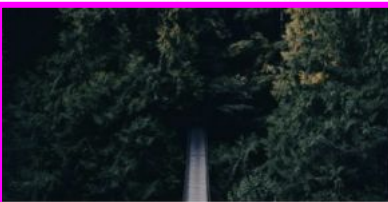
Titulo do video



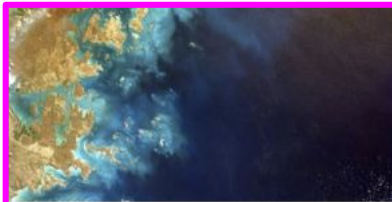
Titulo do video



Titulo do video



Titulo do video



Titulo do video



Titulo do video

O que deve ser um componente?

- **Não existe uma regra** de quando componentizar uma parte da tela
- Devemos considerar **criar** um componente quando:
 - Layout repetido
 - Código muito grande e/ou confuso
 - Queremos dar um nome significativo à uma parte da interface



Componentes em React

Labenu_



Componentes

- Na prática, componentes em React são **funções** (por enquanto) com algumas **regras** específicas:
 - Primeira letra do nome maiúscula
 - Deve retornar um JSX (com um único pai)

```
1 function MeuComponente() {  
2   return <div>  
3     <h1>Meu primeiro componente!</h1>  
4     <p>Este é o meu primeiro componente React</p>  
5   </div>  
6 }
```



Componentes

- Na prática, componentes em React são **funções** (por enquanto) com algumas **regras** específicas:
 - **Primeira letra do nome maiúscula**
 - Deve retornar um JSX (com um único pai)

```
1 function MeuComponente() {  
2   return <div>  
3     <h1>Meu primeiro componente!</h1>  
4     <p>Este é o meu primeiro componente React</p>  
5   </div>  
6 }
```



Componentes

- Na prática, componentes em React são **funções** (por enquanto) com algumas **regras** específicas:
 - Primeira letra do nome maiúscula
 - **Deve retornar um JSX (com um único pai)**

```
1 function MeuComponente() {  
2   return <div>  
3     <h1>Meu primeiro componente!</h1>  
4     <p>Este é o meu primeiro componente React</p>  
5   </div>  
6 }
```

Vamos ver na prática! 



Usando um Componente

- Para colocar o componente na tela, chamamos ele em um **componente pai**, dentro do nosso JSX
- Quando colocamos um componente A dentro de um componente B, falamos que o componente A é **filho** do componente B



Usando um Componente 🌈

- Lembrando que o componente **App.js**, criado por **padrão** quando criamos um app React, **é o pai de todos os outros componentes**



Usando um Componente

- Para **chamar** o componente, usamos uma sintaxe semelhante à do **HTML**
- Nome deve ser mantido, **com a letra maiúscula**

App.js

```
1 function App() {  
2   return <div>  
3     <MeuComponente></MeuComponente>  
4   </div>  
5 }
```

MeuComponente.js

```
1 function MeuComponente() {  
2   return <div>  
3     <h1>Meu Primeiro Componente</h1>  
4     <p>Este é o meu primeiro componente React</p>  
5   </div>  
6 }
```



Self-Closing Tag 🐵

- Quando um componente não possui nada entre a abertura e o fechamento de sua tag, é preferível que se use a sintaxe **self-closing**

App.js

```
1 function App() {  
2   return <div>  
3     <MeuComponente />  
4   </div>  
5 }
```

MeuComponente.js

```
1 function MeuComponente() {  
2   return <div>  
3     <h1>Meu Primeiro Componente</h1>  
4     <p>Este é o meu primeiro componente React</p>  
5   </div>  
6 }
```

Vamos ver na prática! 🔬



Separando em Arquivos

- É permitido ter mais de um componente em um mesmo arquivo (afinal, eles são só funções)
- No entanto, é uma boa prática criar **um arquivo por componente**
- Podemos comunicar os arquivos por meio de **imports** e **exports**



Separando em Arquivos

- O arquivo deve ter o **mesmo nome** do componente que ele guarda
- A organização de pastas é livre, mas recomenda-se a criação de uma pasta chamada **components**, que guarde todos os componentes criados
- Essa pasta pode ter subpastas, de acordo com o que fizer mais sentido para cada um :)



Criando um Componente Separado

- Todo arquivo que possui um componente deve **importar o React no topo**
- Devemos **exportar** o componente (antes do nome)

MeuComponente.js

```
1 import React from 'react'
2
3 export function MeuComponente() {
4   return <div>
5     <h1>Meu primeiro componente!</h1>
6     <p>Este é o meu primeiro componente React</p>
7   </div>
8 }
```



Usando um Componente Separado

- Precisamos **importá-lo** no arquivo onde o usamos
- Fazemos isso por meio da palavra **import**
 - Atenção para as **{chaves}** em volta do nome do componente

App.js

```
1 import React from 'react'
2 import { MeuComponente } from './components/MeuComponente'
3
4 function App() {
5   return <div>
6     <MeuComponente />
7   </div>
8 }
```

Vamos ver na prática! 





Exercício 1

- Usando o LabeTube que foi transferido para um app React, crie um componente que representa o **Card de Vídeo**
- À partir de agora, o card de vídeo será chamado **apenas** por meio deste componente





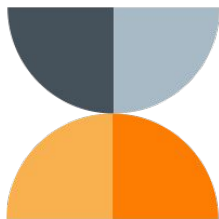
Exercício 2

- Crie um componente que representa um **item** da Lista do Menu
- Renderize ele 6 vezes dentro da lista do App.js, **mantendo os textos repetidos**

Início
Em alta
Inscrições

Originais
Histórico





Pausa para relaxar 🧘

10 min

- Componentes são **funções** que retornam um **JSX**
- Usar componente:
 - `<MeuComponente></MeuComponente>`
 - `<MeuComponente />`
- Criamos um arquivo com o componente
 - **Exportar:** `export` ou `export default`
 - **Importar:**
 - `import {MeuComponente} from './path'`
 - `import MeuComponente from './path'` (no caso do `export default`)



Props

Labenu_



Componentes

- Já vimos que componentes são funções que renderizam HTML na tela

Entrada
(**Props**)



Saída
(**JSX**)

- Frequentemente, vamos querer que o que sai na tela **dependa de alguma informação**



[Início](#)[Em alta](#)[Inscrições](#)[Originais](#)[Biblioteca](#)[Histórico](#)

Título do video



Título do video



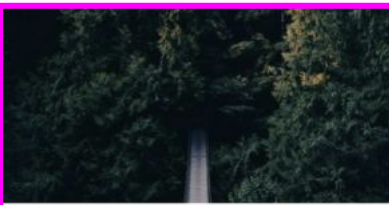
Título do video



Título do video



Título do video



Título do video



Título do video



Título do video

Itens do menu
se repetem,
mudando
somente o texto

Cards de vídeo se
repetem, mudando a
imagem e o texto

Props

- Componentes pais podem passar **propriedades** para os componentes filhos
- Chamamos essas propriedades de **props**
- Elas podem assumir **qualquer tipo** de valor que vimos em Javascript
 - Strings, números, arrays, objetos, funções, componentes, etc...



Passando Props 📁

- A sintaxe é a mesma dos atributos HTML
 - Damos um nome arbitrário
 - Passamos o valor entre {chaves}

```
1 import React from 'react'
2 import { MeuComponente } from '../components/MeuComponente'
3
4 function App() {
5   return <div>
6     <MeuComponente texto={'Testando props'} />
7   </div>
8 }
```



Passando Props

- **A sintaxe é a mesma dos atributos HTML**
 - Damos um nome arbitrário
 - Passamos o valor entre {chaves}

```
1 import React from 'react'
2 import { MeuComponente } from '../components/MeuComponente'
3
4 function App() {
5   return <div>
6     <MeuComponente texto={'Testando props'} />
7   </div>
8 }
```



Passando Props 📁

- A sintaxe é a mesma dos atributos HTML
 - **Damos um nome arbitrário**
 - Passamos o valor entre {chaves}

```
1 import React from 'react'
2 import { MeuComponente } from '../components/MeuComponente'
3
4 function App() {
5   return <div>
6     <MeuComponente texto={'Testando props'} />
7   </div>
8 }
```



Passando Props 📁

- A sintaxe é a mesma dos atributos HTML
 - Damos um nome arbitrário
 - **Passamos o valor entre {chaves}**

```
1 import React from 'react'
2 import { MeuComponente } from '../components/MeuComponente'
3
4 function App() {
5   return <div>
6     <MeuComponente texto={'Testando props'} />
7   </div>
8 }
```



Usando Props

- Quando um componente filho recebe props, temos um **argumento** na função que o define
- Esse argumento é um objeto chamado **props** e contém informações **passadas pelo componente pai**
- O valor passado pelo pai está **disponível dentro do objeto** e pode ser acessado pelo nome da prop passada



Usando Props 🎉

```
1 import React from 'react'
2
3 export function MeuComponente(props) {
4   return <div>
5     <h1>Meu primeiro componente!</h1>
6     <p>Este é o meu primeiro componente React</p>
7     <p>{props.texto}</p>
8   </div>
9 }
```



Usando Props

- Quando um componente filho recebe props, temos um **argumento** na função que o define

```
1 import React from 'react'
2
3 export function MeuComponente(props) {
4   return <div>
5     <h1>Meu primeiro componente!</h1>
6     <p>Este é o meu primeiro componente React</p>
7     <p>{props.texto}</p>
8   </div>
9 }
```



Usando Props

- Esse argumento é um objeto chamado **props** e contém informações **passadas pelo componente pai**

```
1 import React from 'react'
2
3 export function MeuComponente(props) {
4   return <div>
5     <h1>Meu primeiro componente!</h1>
6     <p>Este é o meu primeiro componente React</p>
7     <p>{props.texto}</p>
8   </div>
9 }
```

{ texto: 'Testando props' }



Usando Props

- O valor passado pelo pai está **disponível dentro do objeto** e pode ser acessado pelo **nome da prop** passada

```
1 import React from 'react'
2
3 export function MeuComponente(props) {
4   return <div>
5     <h1>Meu primeiro componente!</h1>
6     <p>Este é o meu primeiro componente React</p>
7     <p>{props.texto}</p>
8   </div>
9 }
```

Vamos ver na prática! 



Regra Importante

- Props são *read-only* \Rightarrow **Somente Leitura**
- Componentes filhos não podem **nunca** alterar o objeto de props



Regra Importante



- Props são *read-only* ⇒ **Somente Leitura**

- Componentes filhos não podem **nunca** alterar o objeto de props





Exercício 3

- Adicionar **props** aos componentes criados nos exercícios 1 e 2
- Eles devem conseguir receber títulos, imagens e textos diferentes



Inicio

Em alta

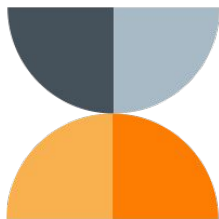
Inscrições

Originais

Biblioteca

Histórico





Pausa para relaxar 🤔

5 min

- Para reutilizar um componente, usamos as **props**
- Props são dados passados de componente pai para componente filho e são **read-only**
- **Passando props:** nome arbitrário e valor entre chaves
- **Recebendo props:** objeto props, propriedade com o nome arbitrário escolhido



styled-components

Labenu_



Styled Components

- Já temos HTML e JS juntos, só o CSS permanece separado
- **Trabalhoso manter nomes de classes**
 - O React **unifica** todo o **código de CSS** em um arquivo só, então, temos que garantir que **não** iremos **repetir** os nomes
 - Isso é chato (ou até mesmo impossível)



Styled Components

- **Styled Components** é uma lib que garante que o CSS de cada componente seja totalmente restrito àquele componente
- Existem outras libs, mas escolhemos essa por ser **famosa no mercado**
- Para **instalar**, rodar: `npm install styled-components`



Exemplo sem styled-components

```
1 function App() {  
2   return (  
3     <div>  
4       <h1 className="titulo-vermelho">Meu Título</h1>  
5     </div>  
6   )  
7 }
```

Meu Título

Vamos ver na prática! 



Exemplo com styled-components

```
1 import styled from 'styled-components'
2
3 const RedTitle = styled.h1`
4   color: red;
5 `
6
7 function App() {
8   return (
9     <div>
10       <RedTitle>Meu Título</RedTitle>
11     </div>
12   )
13 }
```

Meu Título

Vamos ver na prática! 



Exemplo com styled-components

```
1 import styled from 'styled-components'
2
3 const RedTitle = styled.h1`
4   color: red;
5 `
6
7 const TitleContainer = styled.div`
8   display: flex;
9   align-items: center;
10  height: 60px;
11  background-color: blue;
12  padding-left: 10px;
13 `
14
15 function App() {
16   return (
17     <TitleContainer>
18       <RedTitle>Meu Título</RedTitle>
19     </TitleContainer>
20   )
21 }
```

Meu Título

Vamos ver na prática! 





Exercício 4


- Realize a estilização do footer do LabeTube usando styled components ao invés do CSS

```
1 footer {  
2   background: #333b3e;  
3   color: white;  
4   position: fixed;  
5   bottom: 0;  
6   width: 100%;  
7   display: flex;  
8   padding: 0 10px;  
9 }
```



Dicas



- Instalem essa extensão no de vocês VSCode. Ela ajuda muito a usar styled-components!



styled
components

vscode-styled-components

jpoissonnier.vscode-styled-components

Julien Poissonnier |  352.536 |  | Repository | License | 0.0.29

Syntax highlighting for styled-components

[Disable ▼](#) [Uninstall](#) *This extension is enabled globally.*



Dicas

- Nós não passaremos por cada detalhezinho de como fazer coisa X ou Y usando styled components
- A sintaxe é bem parecida com o CSS comum e, se você não souber algo, basta pesquisar no google **"Como fazer TAL COISA com styled components"**
- Ainda assim, deixamos um compilado de dicas pronto pra ajudar vocês [nesse documento](#)



Resumo

Labenu_



Resumo

- Componentes são **funções** que retornam um **JSX**
- Usar componente:
 - `<MeuComponente></MeuComponente>`
 - `<MeuComponente />`
- Criamos um arquivo com o componente
 - **Exportar:** `export` ou `export default`
 - **Importar:**
 - `import {MeuComponente} from './path'`
 - `import MeuComponente from './path'` (no caso do `export default`)



Resumo

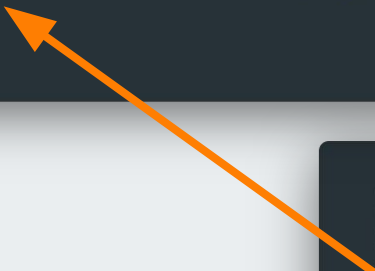
- Para reutilizar um componente, usamos as **props**
- Props são dados passados de componente pai para componente filho e são **read-only**
- **Passando props:** nome arbitrário e valor entre chaves
- **Recebendo props:** objeto props, propriedade com o nome arbitrário escolhido



Resumo



```
1 import React from 'react'
2 import { MeuComponente } from '../components/MeuComponente'
3
4 function App() {
5   return <div>
6     <MeuComponente texto={'Testando props'} />
7   </div>
8 }
```



```
1 import React from 'react'
2
3 export function MeuComponente(props) {
4   return <div>
5     <h1>Meu primeiro componente!</h1>
6     <p>Este é o meu primeiro componente React</p>
7     <p>{props.texto}</p>
8   </div>
9 }
```



Resumo

- **Styled Components** é uma das libs que permitem que todas as partes do site estejam em **um lugar só**
- Precisamos tomar cuidado com a **sintaxe** e as **crases**!
- Caso não saiba como fazer algo utilizando styled-components, não se desespere! **PESQUISE** <3



Dúvidas? 🧐

Labenu_





Obrigado(a)!