Renderização Condicional

Labenu_

O que vamos ver hoje?

- Coerção de Booleanos
- Expressões x Instruções
- Expressões Condicionais
- Renderização Condicional

Coerção de Booleanos

Labenu_

Relembrando... 

- Um valor booleano é algo que só pode assumir um dos dois valores: **true** ou **false**

- Podemos combinar esses valores usando operadores lógicos ⇒ && e ||

- Comparações (===, !==, <, <=, >, >=) sempre geram um valor booleano

Coerção ✅

- Normalmente usamos booleanos em condicionais
- Mas o que acontece se tentarmos colocar um valor **não booleano** em uma condicional? 😱
- O javascript **converte** o valor para booleano na hora!
- Chamamos esse processo de **coerção**

Tipos de Coerção ✅

VALOR	COERÇÃO
0	false
1,2,3,4,...	true
null	false
undefined	false
NaN	false
Strings vazias ("")	false
"Strings não-vazias"	true
Todo o resto	true

- Valores convertidos para **false** são chamados **falsy**
- Valores convertidos para **true** são chamados **truthy**

Exemplo

Expressions x Statements

Labenu_

Expressions 1️⃣

- Expressões e operadores resultam em um valor ou se tornam um valor.
- Um jeito fácil de lembrar: algo que pode ser atribuído a uma variável ou aparecer dentro do console.log()

- Exemplos:
  - Um valor em si (`"texto"`, `2`, `{name: "João"}`, `[1,2,3]`)
  - Operações (`3 * 5`, `"Hello" + "World"`)
  - Chamadas de Funções

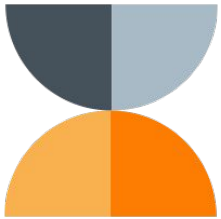
Statements 🔁

- Representam ações.
- Um jeito fácil de lembrar: algo que não pode ser atribuído a uma variável
- Exemplos:
  - Condicionais comuns (`if`, `else`)
  - Loops (`while`, `for`)
  - Atribuições (const num = 1)

"Problema" ⚠️

- Em algumas situações, só podemos usar **Expressões**
  - JSX (return do método render())
  - Template Strings
  - Atribuição de Variáveis

- Além disso, condicionais são muito **verbosas**
- E, às vezes, queremos condicionais mais simples.



Pausa para relaxar 😴

5 min

- **Coerção** transforma valores não booleanos em booleanos de acordo com uma tabela

- **Expressões** e operadores resultam em um valor ou se tornam um valor.

- **Declarações** representam ações.

Expressões Condicionais

Labenu_

Expressões Condicionais 💡

- **Expressões Condicionais** permitem a definição de um valor condicionalmente

- São expressões, ou seja, avaliadas para um **único valor**

- Resolvem casos específicos das condicionais tradicionais

- Veremos dois tipos: **Ternários** e **Curto Circuito**

Ternário 🔨

- O ternário permite definir um **valor** a partir de uma condição

- É como "transformar" um **if/else** em uma expressão

- Definimos três coisas (daí o nome)
  - Condição
  - Expressão verdadeira
  - Expressão falsa

Ternário - Sintaxe 🔨

```
CONDICAO ? EXPR_SE_VERDADEIRO : EXPR_SE_FALSO
```

Se valor não for booleano, ocorre coerção

Expressão toda irá assumir somente **uma** das duas expressões, de acordo com a condição

Vamos ver na prática! 

Curto Circuito 🔧

- Ao fazer operações com operadores lógicos **&&** e **||**, a linguagem "para de executar" assim que **já sabe o resultado**

- Isso permite que usemos esses operadores para criar expressões condicionais

Curto Circuito - && 🔧

- O operador && retorna true somente quando **ambos os valores são true**

- Portanto, se o primeiro valor é ***falsy***, não é necessário nem olhar para o segundo valor

- Assim, é possível usá-lo para executar uma expressão condicionalmente somente quando algum valor específico for ***truthy***

Curto Circuito - || 🔧

- O operador || retorna false somente quando **ambos os valores são false.**

- Portanto, se o primeiro valor é ***truthy***, não é necessário nem olhar para o segundo valor

- Assim, é possível usá-lo para **executar uma expressão** condicionalmente somente quando algum valor específico for ***falsy***

Curto Circuito - Exemplos 🔧

```
let minhaVariavelFalsy = null
let minhaVariavelTruthy = 2


minhaVariavelTruthy && minhaVariavelFalsy // retorna a segunda expressão
minhaVariavelFalsy && minhaVariavelFalsy // retorna a primeira expressão


minhaVariavelTruthy || minhaVariavelFalsy // retorna a primeira expressão
minhaVariavelFalsy || minhaVariavelFalsy //  retorna a segunda expressão
```

Vamos ver na prática! 🔬

Curto Circuito - Exemplos 

EXPR_1 && EXPR_2

EXPR_1	EXPR_1 && EXPR_2
`truthy`	**EXPR_2**
`falsy`	**EXPR_1**

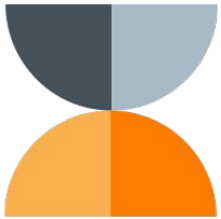
EXPR_1 || EXPR_2

EXPR_1	EXPR_1 || EXPR_2
`truthy`	**EXPR_1**
`falsy`	**EXPR_2**

Vamos ver na prática! 

Pausa para relaxar 😴

10 min



- **Ternários** ⇒ Permitem fazer uma condicional em forma de **expressão**. Não retorna a condição.
- **Curto Circuito** ⇒ propriedade que permite usar operadores **&&** e **||** para fazer condicionais.

Renderização Condicional

Labenu_

Conceito 🔀

- Queremos modificar o que é renderizado na tela com base em **parâmetros**, normalmente, vindos de **estado** ou **props**

- Componentes retornam **JSX** e o que aparece na tela é determinado por eles

- Vamos usar **condicionais** Javascript para determinar o JSX a ser retornado

Estratégia 🔀

- Existem várias formas de retornar JSX diferentes com base nas diferentes formas que temos de fazer **condicionais em Javascript**

- Vamos ver algumas:
  - if/else
  - Switch case
  - Expressões condicionais (ternários e &&)

Exercício 1

- Vamos criar um protótipo de rede social com 2 páginas:
  - Login
  - Home

- Em cada uma delas, vamos controlar a renderização dos elementos usando uma das possíveis estratégias que vimos hoje

App.js

- No componente App, vamos usar **if/else** para decidir se mostramos a tela de Login ou a tela de Home

- Cada uma dessas telas deve ser um componente

Home.js

- No componente Home, vamos usar **Switch case** para definir qual seção devemos mostrar: Posts ou Mensagens

- Cada uma dessas seções deve ser um componente

Mensagens.js

- No componente Mensagens, vamos usar um **ternário** para decidir qual mensagem deve ser mostrada

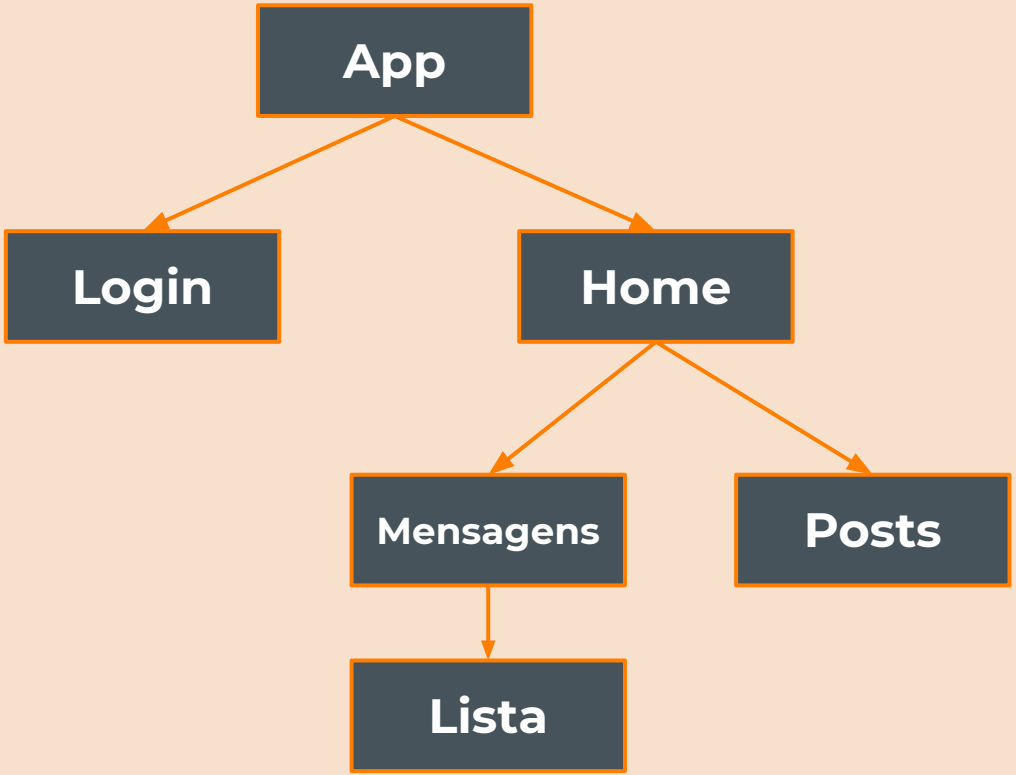
- Se **não tiver** nenhuma mensagem:
  - "Não há novas mensagens"

- Se **tiver** mensagens:
  - "Quantidade de mensagens"

Mensagens.js

- Ainda no componente Mensagens, vamos usar um **&&** para mostrar um componente ListaMensagens caso a lista de mensagens seja válida

Estrutura Exercício 1



Resumo

Labenu_

Resumo 📓

- **Não conseguimos** usar if/else (**Statement**) dentro do JSX

- Para colocar condições nessa parte do código, usamos:
  - **Ternários:** funciona como if/else
  - **Curto Circuito:** frequentemente usado para verificar se um dado realmente existe onde nós esperamos

- As verificações podem ser feitas com valores não-booleanos que serão convertidos pelo JS (**Coerção de Booleanos**)

Dúvidas? 

Labenu_

Labenu_

Obrigado(a)!