

Semana 5 - Aula 3

Renderização Condicional

FUTURE4

Sumário

O que vamos ver hoje? 🙄

- Hoje nossa aula será sobre **Renderização Condicional**
- Basicamente, é um conceito abrangente que envolve o uso de condicionais em renderização de componentes do React
- Antes de ver isso com detalhe, nós vamos aprofundar um pouco em **Condicionais**

O que vamos ver hoje? 🙄

- 3 tópicos:
 - Template Strings
 - Expressões Condicionais
 - Renderização Condicional

Template Strings

Strings, strings e mais strings 🧵

- Vocês já tiveram dificuldades em criar uma *string* muito **complexa**?
- Lembrem-se das aulas em que usávamos HTML cru para criar listas.

```
15  |   const pessoa = { nome: "Goli" }  
16  |   document.getElementById("meu-id").innerHTML  
17  |   |   += "<div>" + "<p>" + pessoa.nome + "</p>" + "</div>"  
18  |
```

Strings, strings e mais strings 🧵

- Existe uma maneira simples de criar *strings* complexas: **Template Strings**
- Antes de ensinar eles, vamos nos lembrar de como criar uma *string* em JS:

```
1  
2  const meuNome = "João V. Golias"  
3  
4  const nomeDoDarvas = 'Pedro V. Darvas'  
5
```

Strings, strings e mais strings 🧵

- Para usar caracteres especiais, temos que colocar o **\ (barra invertida)** na frente.

```
6  const stringComAspas = " Olá, me chamo \" Goli \" "  
7  
8  const stringComBarraInvertida = " Uma barra invertida: \\ Agora duas: \\\\ "  
9  
10 const stringComEnter = "Quebra de linha faz assim: \n"  
11
```

- Caracteres reservados:
 - **\n**: "new line"
 - **\b**: "backspace"
 - **\r**: "return"

Vamos ver na prática! 🚫

Template Strings 🏯

- **Template String** devem, sempre, ser inicializadas com **back-ticks** (` `)
- A vantagem da **Template String** são duas:
 - A string será criada com a exata **formatação** que colocarmos
 - Conseguimos nos referenciar **a expressões de JS**, desde que coloquemos dentro de **\${ }**

Template Strings

```
1  const meuNome = "Golias"
2  const minhaIdade = 23
3
4  const dizOla = () => {
5    |   return "Olá"
6    | }
7
8  const stringFinal =
9    | ` ${dizOla()}, meu nome é ${meuNome} e tenho ${minhaIdade} anos de idade.
10   |
11   |   Oie, moro no fim da string. hehe
12   |   |   E eu tenho uma tab no início
13   | `
14
15  console.log(stringFinal)
```

Vamos ver na prática! 

FUTURE 

Exercício 1 - *Template Strings*

Escreva uma função que receba dois números e imprima:

"INFORMAÇÕES:

- O maior número é: <numero>.
- A diferença entre eles é: <numero>."

Expressões Condicionais

Ternários

- Ternário é uma estrutura, **muito comum nas linguagens de programação**, que permite fazer condicionais em uma só linha

7

8

9

```
BOOLEANO ? VALOR_SE_VERDADEIRO : VALOR_SE_FALSO
```

Ternários 🙋

- Um ternário é uma **expressão de Javascript**, isso implica que uma variável pode assumir o seu valor

```
13  let minhaIdade = 23
14  let ehMaisVelhoDoQue20 = minhaIdade > 20 ? "Sim" : "Não"
15
16  console.log(ehMaisVelhoDoQue20)
17
18  minhaIdade = 10
19  ehMaisVelhoDoQue20 = minhaIdade > 20 ? "Sim" : "Não"
20
21  console.log(ehMaisVelhoDoQue20)
22
```

Vamos ver na prática! 🚫

FUTURE4

Exercício 2 - *Template Strings E Ternários*

Escreva um programa que crie os emails a serem enviados para clientes de uma empresa. Siga o exato formato abaixo e tome cuidado com o uso do "feminino" e "masculino"

“Olá, <nome>

Seja bem-vinda(o) a nossa plataforma.

Estamos muito felizes em poder recebê-la(o).

Aguarde contato para mais informações.”

Pausa para relaxar

5 min

Coerção de Condicionais

- As expressões if/else esperam receber em seus parênteses variáveis com os valores *boolean*
- O que acontece se colocarmos um valor que não seja *boolean*?
- Ocorre o que chamamos de coerção (ou **cast**): o Javascript, em tempo de execução, tenta converter a variável para um *boolean*; e, aí, faz a verificação

Coerção de Condicionais

- Tabela de Coerção para Boolean

VARIÁVEL	CAST PARA BOOLEAN
0	<i>false</i>
null	<i>false</i>
undefined	<i>false</i>
NaN	<i>false</i>
Strings vazias ("")	<i>false</i>
TODO O RESTO	true

Vamos ver na prática! 

FUTURE 

Operador &&

- Esse operador é lógico. O resultado dele é **true**; se, e somente se, as duas variáveis forem **true**

```
1  true && true === true
2  true && false === false
3  false && true === false
4  false && false === false
5
```

Operador &&

- Se repararmos bem, tem um padrão nas respostas.
- Sempre que o valor **na esquerda** do operador for **true**, a resposta da operação, é o valor na direita


```
1  true && true === true
2  true && false === false
```

Operador &&

- Sempre que o valor **na esquerda** do operador for **false**, a resposta da operação será o próprio valor na esquerda

```
3  false && true == false
4  false && false == false
```

Operador &&

- Como eu disse, o operador && é usado entre dois boolean
- **O que acontece se tentarmos usar ele com algum valor que não é boolean?**
- Ocorre a coerção! Ele faz um **cast** para boolean e vale as regras citadas
- **Para que usar isso?** 

Vamos ver na prática! 

Exercício 3 - *Expressões condicionais*

Se em todos os exemplos atrás que eu dei, ao invés do operador **&&**, tivesse o operador **||** ? Quais seriam os resultados esperados?

```
1  let resultado = true || true
2  resultado = true || false
3  resultado = false || true
4  resultado = false || false
5
6  resultado = true || 1
7  resultado = false || 1
8  resultado = 0 || 1
9  resultado = 1 || 12
10
```

Pausa para relaxar

10 min

Renderização Condicional

Renderização Condicional 📦

- O conceito de renderização condicional é simples
 - É uma maneira de modificar qual componente (ou uma parte dele) será renderizado dependendo das informações no site
- A renderização condicional é feita com expressões condicionais:
 - Blocos if/else
 - Ternários
 - Operador lógico &&

Renderização Condicional 📁

- Vamos ver 3 casos importantes
 - Retornando componentes diferentes
 - Usando ternários
 - Usando o comparador lógico &&

1. Retornando componentes diferentes

- O princípio deste caso é uma função **retornar** componentes de **JSX** diferentes
- Usando blocos **if/else**

```
26 function MeuComponente(props) {  
27   if(props.algumaCondicao) {  
28     return (<div> ENTREI NO IF </div>)  
29   } else {  
30     return (<div> NÃO ENTREI NO IF </div>)  
31   }  
32 }  
33
```

1. Retornando componentes diferentes

- **Exemplo:**

- Vamos fazer um site simples que tenha um botão de "Logar" e uma mensagem acima.
- Quando o usuário clicar em "Logar", ele deve ser encaminhado a uma página específica. Ela tem uns botões específicos para o usuário logado

1. Retornando componentes diferentes

- **Exemplo:**

Por favor, logue-se!

Login

Bem vindo de volta!

Feed

Lista de Amigos

Sair

2. Usando ternários

- Conseguimos usar ternários ao invés de **if/else**
- Neste caso, eles são chamados de **inline blocks**
 - Porque um **bloco** foi transformado em uma sintaxe que usa só uma linha

```
5  function MeuComponente(props) {  
6    |   return props.algumaCondicao ? (<div> SIM </div>): ( <div> NÃO </div>)  
7    | }  
8
```

2. Usando ternários

- **Exemplo:**

- Vamos refazer o exemplo anterior, mudando o bloco if/else pelo ternário

VAMOS FAZER JUNTOS! 

3. Usando operador &&

- Como já vimos o operador && é muito útil
- Podemos usá-lo para verificar se algo é **null/undefined** antes de usar a variável em si
- Nesta lógica, podemos fazer uma verificação de alguma variável; e, só então, renderizar um componente

3. Usando operador &&

- **Exemplo:**

- Na nossa rede social que começamos a fazer, os usuários podem enviar mensagem um para os outros.
- Se o usuário tiver novas mensagens para ver, vamos mostrá-lo, ao se logar, um aviso:

Você possui X mensagens não lidas

Resumo

- Existe uma maneira bem fácil de se criar strings complexas: **template strings**
 - Elas são declaradas entre back-ticks/crases (` `)
- Se tentarmos fazer operações com variáveis de tipos diferentes, o Javascript faz a conversão entre valores. É o que chamamos de **coerção**

Resumo

VARIÁVEL	CAST PARA BOOLEAN
0	<i>false</i>
null	<i>false</i>
undefined	<i>false</i>
NaN	<i>false</i>
Strings vazias ("")	<i>false</i>
TODO O RESTO	<i>true</i>

Resumo

- **Renderização condicional:** usar expressões condicionais para mostrar elementos
- **3 casos marcantes:**
 - Retornando componentes diferentes
 - Ternários
 - Operador &&

Dúvidas?

Obrigado!