

Executar os processos de codificação, manutenção e documentação de aplicativos computacionais para desktop

SENAC PE

Thiago Nogueira

Instrutor de Educação Profissional

React: Renderização condicional

Renderização Condicional

Como funciona

- No React, **renderização condicional** significa mostrar (ou esconder) algo na tela com base em certas condições.
- Por exemplo, você pode querer mostrar uma mensagem como "Bem-vindo de volta!" se um usuário estiver logado, mas mostrar uma mensagem "Por favor, faça login" se ele não estiver logado.
- É como tomar decisões no seu aplicativo: "Se isso acontecer, mostre isso; caso contrário, mostre aquilo."

Renderização Condicional

Como funciona

- O React permite que você use a lógica JavaScript para renderizar condicionalmente coisas dentro dos seus componentes.
- Isso significa que você pode usar **if** , **else** , **operadores ternários** ou até mesmo funções JavaScript para decidir o que deve aparecer na interface do usuário (UI). Pode ser do tipo:
 - 1. Usando instruções if
 - 2. Usando o operador ternário
 - 3. Usando && (E lógico)
 - 4. Usando funções para renderização condicional
 - 5. Renderizando vários elementos com base em condições
 - 6. Renderização condicional com nulo

Renderização Condicional

Usando instruções if

- Em JavaScript, instruções **if** permitem que você execute algum código somente se uma condição for verdadeira.
- Você pode usar isso dentro de um componente React para decidir o que renderizar.

```
function Item({ name, isPacked }) {  
  if (isPacked) {  
    return <li className="item">{name} ✓</li>;  
  }  
  return <li className="item">{name}</li>;  
}  
  
export default function PackingList() {  
  return (  
    <section>  
      <h1>Sally Ride's Packing List</h1>  
      <ul>  
        <Item  
          isPacked={true}  
          name="Space suit"  
        />  
        <Item  
          isPacked={true}  
          name="Helmet with a golden leaf"  
        />  
        <Item  
          isPacked={false}  
          name="Photo of Tam"  
        />  
      </ul>  
    </section>  
  );  
}
```



Sally Ride's Packing List

- Space suit ✓
- Helmet with a golden leaf ✓
- Photo of Tam

Renderização Condicional

Usando operador ternário

- Outra maneira de fazer renderização condicional é com o operador ternário. É uma abreviação para **if...else**. Eis como funciona:

Ao invés disso:

```
if (isPacked) {  
  return <li className="item">{name} ✓ </li>;  
}  
return <li className="item">{name}</li>;
```

Escreva isso:

```
return (  
  <li className="item">  
    {isPacked ? name + ' ✓' : name}  
  </li>  
)
```

Renderização Condicional

Usando &&

- O operador **&&** renderiza conteúdo somente quando uma condição é verdadeira, sem um caso **else**.
- Eis um exemplo:

```
function Item({ name, isPacked }) {  
  return (  
    <li className="item">  
      {name} {isPacked && '✔'}  
    </li>  
  );  
}  
  
export default function PackingList() {  
  return (  
    <section>  
      <h1>Sally Ride's Packing List</h1>  
      <ul>  
        <Item  
          isPacked={true}>  
          name="Space suit"  
        </Item>  
        <Item  
          isPacked={true}>  
          name="Helmet with a golden leaf"  
        </Item>  
        <Item  
          isPacked={false}>  
          name="Photo of Tam"  
        </Item>  
      </ul>  
    </section>  
  );  
}
```



Sally Ride's Packing List

- Space suit ✔
- Helmet with a golden leaf ✔
- Photo of Tam

Renderização Condicional

Usando funções

- O operador **&&** renderiza conteúdo somente quando uma condição é verdadeira, sem um caso **else**.
- Eis um exemplo:

```
import React from 'react';

const ConditionalRenderingExample = () => {
  const isLoggedIn = true;

  const renderMessage = () => {
    if (isLoggedIn) {
      return <h1>Welcome back!</h1>;
    } else {
      return <h1>Please log in.</h1>;
    }
  };

  return (
    <div>
      {renderMessage()}
    </div>
  );
};

export default ConditionalRenderingExample;
```


Renderização Condicional

Condicional com nulo

- Às vezes, você não quer mostrar nada quando uma condição é falsa.
- No React, você pode retornar **null** para renderizar nada.
- Eis um exemplo:
- Se **showWarning** for falso, nada será exibido para essa parte do componente.

```
import React from 'react';

const ConditionalRenderingExample = () => {
  const showWarning = false;

  return (
    <div>
      <h1>Dashboard</h1>
      {showWarning ? <p>Warning: Something went wrong!</p> : null}
    </div>
  );
};

export default ConditionalRenderingExample;
```

React: Manipulando eventos no React

Manipulando eventos

Como funciona

- No desenvolvimento web, eventos são ações que ocorrem **quando o usuário interage com seu aplicativo**.
- Exemplos comuns de eventos incluem clicar em um botão, passar o mouse sobre um elemento ou enviar um formulário.
- O React permite que você manipule esses eventos de forma estruturada usando manipuladores de eventos.

Manipulando eventos

Principais diferenças React vs. HTML/JS normal

- O React usa camelCase para nomes de eventos. Por exemplo, `onClick` no React em vez de no HTML regular.
- Em vez `onclick` de strings, no React, você passa uma função como o manipulador de eventos. Em HTML, você pode escrever algo como `onclick="doSomething()"`, mas no React, você passa a função diretamente, sem aspas.
- Essas diferenças tornam o tratamento de eventos do React mais poderoso e flexível.

Manipulando eventos

Clique de botão



```
import React from 'react';

function App() {
  // This function will run when the button is clicked
  function handleClick() {
    alert('Button was clicked!');
  }

  return (
    <div>
      <button onClick={handleClick}>Click Me</button>
    </div>
  );
}

export default App;
```

Analizando o exemplo:

- **A função handleClick:** Esta é a função que o React executará sempre que o botão for clicado. Neste caso, a função mostra um alerta pop-up que diz "O botão foi clicado!"
- **O evento onClick:** Anexamos o evento **onClick** diretamente ao botão. Sempre que o botão é clicado, o evento dispara a função **handleClick**, que então mostra a mensagem de alerta.

Manipulando eventos

Passando informações

- Pode haver situações em que você queira manipular diferentes ações com base em qual botão é clicado.
- Por exemplo, imagine que você tem vários botões na página e quer saber exatamente em qual botão o usuário clicou.
- Para manipular isso, você pode passar argumentos (ou valores) para o manipulador de eventos.

Manipulando eventos

Passando informações



```
import React from 'react';

function App() {
  // This function accepts a 'buttonName' and shows it in the alert
  function handleClick(buttonName) {
    alert(`${buttonName} button was clicked!`);
  }

  return (
    <div>
      <button onClick={() => handleClick('First')}>First Button</button>
      <button onClick={() => handleClick('Second')}>Second Button</button>
    </div>
  );
}

export default App;
```

Analizando o exemplo:

- **A função handleClick:** Ela aceita um argumento chamado **buttonName**. Quando o botão é clicado, esta função mostra qual botão foi clicado exibindo uma mensagem.
- **Passando Argumentos para o Manipulador de Eventos:** Para passar o nome do botão para a função **handleClick**, usamos uma função de seta dentro do evento **onClick**: Isso nos permite passar a string 'First' para a função quando o primeiro botão é clicado, e 'Second' quando o segundo botão é clicado.

Exercícios

Exercícios

Instruções

- Utilize a lista feita no exercício anterior
- Crie um botão para interagir condicionalmente com a lista (O que a função faz fica a seu critério)
- Atualiza o README
- Atualiza o repositório
- Envia o link no teams


*“Ensinar é impregnar
de sentido o que
fazemos a cada
instante”*


Paulo Freire

Obrigado!

Thiago Nogueira

 [linkedin.com/tdn](https://www.linkedin.com/tdn)

 thiago.nogueira@pe.senac.br

 (81) 9 9627-0419