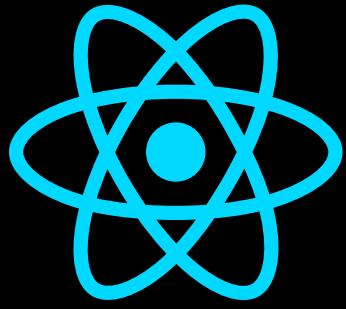


Componentes no React



O que iremos aprender?

- Formato JSX
- Renderização no React
- Renderizando Listas no React
- Renderizando listas - Exercício
- Criando componentes no React
- Criando componentes no React - Exercício
- Importando componentes
- Importando componentes - Exercício
- Propriedades no React
- Propriedades no React - Exercício
- Revisando Conceitos
- Colocando em prática

Formato JSX

JSX é uma extensão de sintaxe usada em React para escrever código que se parece com HTML, mas que é realmente JavaScript por baixo dos panos. Aqui estão algumas regras e convenções básicas para o formato JSX:

Sintaxe Familiar HTML-like: No JSX, você pode escrever tags HTML diretamente em seu código JavaScript, o que facilita a criação de componentes de interface do usuário.

Formato JSX

```
1 // 1. Sintaxe Familiar HTML-like
2 const elementoJSX = <div>Hello, World!</div>;
```

Usando Chaves para Expressões JavaScript: Você pode incorporar expressões JavaScript dentro do JSX usando chaves {}. Por exemplo, {variavel} ou {2 + 2}.

```
1 // 2. Usando Chaves para Expressões JavaScript
2 const variavel = 'Mundo';
3 const elementoComExpressao = <div>Hello, {variavel}!</div>;
```

Formato JSX

Atributos em CamelCase: Ao definir atributos em elementos JSX, você deve usar a convenção de escrita CamelCase. Por exemplo, `className` em vez de `class`, e `onClick` em vez de `onclick`.

```
1 // 3. Atributos em CamelCase
2 const elementoComAtributo = <input type="text" className="input-text" />;
```

No JSX, as tags devem ser explicitamente fechadas, assim como no HTML. Isso significa que todas as tags devem ter uma tag de fechamento correspondente ou devem ser auto-fechadas, se forem tags vazias.

Formato JSX

```
1 // Tag com fechamento explícito
2 const tagComFechamento = <div>Conteúdo</div>;
3
4 // Tag vazia auto-fechada
5 const tagVazia = ;
```

Estilos Inline: Para adicionar estilos inline em elementos JSX, você os define como objetos JavaScript. Por exemplo, { color: 'red', fontSize: '16px' }.

Formato JSX

```
2 // Estilos Inline
3 const estiloInline = {
4   color: 'red',
5   fontSize: '16px',
6 };
7
8 const ElementoComEstiloInline = () => {
9   return <div style={estiloInline}>Texto com estilo inline</div>;
10};
```

Renderização no React

No React, renderização refere-se ao processo de transformar componentes de React em elementos DOM (Document Object Model) que são exibidos na interface do usuário do navegador.

Virtual DOM:

- O React usa uma abstração chamada Virtual DOM para representar a estrutura do DOM em memória.
- O Virtual DOM é uma cópia leve do DOM real e é usado pelo React para efetuar comparações eficientes e determinar quais partes do DOM precisam ser atualizadas.

Renderização no React

Reconciliação:

- Quando ocorre uma mudança no estado ou nas propriedades de um componente, o React inicia um processo chamado reconciliação.
- Durante a reconciliação, o React compara a versão anterior do Virtual DOM com a versão atual, identificando as diferenças entre eles.

Renderização no React

Atualização Seletiva:

- Com base nas diferenças identificadas durante a reconciliação, o React atualiza apenas as partes do DOM que foram alteradas, em vez de re-renderizar todo o DOM.
- Isso torna a renderização no React mais eficiente, especialmente em aplicações complexas com muitos componentes.

Renderização no React

Re-renderização Condicional:

- O React permite a renderização condicional, onde você pode decidir se um componente ou parte dele deve ser renderizado com base em condições específicas.
- Isso é feito usando declarações condicionais, como if, ternário ou métodos como map para renderizar listas condicionalmente.

Renderização no React

```
1 import React from 'react';
2
3 function App() {
4   const mostrarMensagem = true; // Altere para true ou false para ver a renderização condicional
5
6   // Renderização condicional usando operador ternário
7   return (
8     <div>
9       <h1>Renderização Condicional</h1>
10      {mostrarMensagem ? <div>A mensagem está sendo exibida!</div> : null}
11    </div>
12  );
13}
14
15 export default App;
```

Renderização no React - Renderizando Listas

No React, a renderização de listas é um processo pelo qual você exibe elementos de um array de dados na interface do usuário.

Isso é comumente feito usando o método `map()` de arrays em conjunto com JSX para criar elementos de lista.

Aqui está um passo a passo do que é necessário para renderizarmos uma lista no React:

Tenha um array de dados: Primeiro, você precisa ter um array de dados que deseja exibir na lista. Isso pode ser qualquer tipo de dados - nomes, números, objetos, etc.

Renderização no React - Renderizando Listas

Crie um componente React: Em seguida, você precisa criar um componente React onde deseja renderizar a lista.

Use o método map() para iterar sobre o array de dados: Dentro do componente React, você pode usar o método map() do JavaScript para percorrer cada elemento do array de dados.

Retorne elementos JSX dentro da função map(): Para cada elemento do array, retorne um elemento JSX que represente um item da lista. Isso geralmente é feito dentro de um bloco de chaves {} para permitir a expressão JSX em JavaScript.

Renderização no React - Renderizando Listas

Atribua uma chave única a cada elemento da lista: Ao criar elementos de lista dinamicamente com o método `map()`, é importante atribuir uma chave única a cada elemento. Isso ajuda o React a identificar quais elementos foram adicionados, removidos ou modificados durante a atualização da lista.

```
import React from 'react'; // Importa o React

function ListaNomes() { // Cria um componente chamado ListaNomes
  const nomes = ['João', 'Maria', 'José', 'Ana']; // Define um array de nomes

  return ( // Retorna o conteúdo JSX do componente
    <div> {/* Início do elemento div */}
      <h1>Lista de Nomes</h1> {/* Título da lista */}
      <ul> {/* Início da lista não ordenada */}
        {nomes.map(nome => ( // Usa o método map para iterar sobre o array de nomes
          <li key={nome}>{nome}</li> // Para cada nome, cria um elemento <li> com o nome dentro
        ))}
      </ul> {/* Fim da lista não ordenada */}
    </div> // Fim do elemento div
  );
}

export default ListaNomes; // Exporta o componente ListaNomes
```

Renderizando Listas - Exercício

Questão: Crie um componente React chamado `ListaDeUsuarios`. Este componente deve exibir uma lista não ordenada (``) contendo os nomes de usuários fornecidos em uma matriz (`usuarios`). Certifique-se de que cada nome de usuário seja exibido como um item de lista (``).

Instruções:

1. Crie um componente funcional `ListaDeUsuarios`.
2. Dentro do componente, defina uma matriz de nomes de usuários chamada `usuarios`.
3. Use o método `map()` para iterar sobre a matriz `usuarios`.
4. Para cada nome de usuário na matriz, retorne um elemento `` contendo o nome de usuário.
5. Não se esqueça de adicionar uma chave única para cada item de lista.
6. Renderize a lista de usuários dentro de uma lista não ordenada ``.

Criando componentes no React

A componentização no React é um conceito fundamental que permite dividir a interface do usuário em partes menores e reutilizáveis chamadas de componentes. Cada componente encapsula um pedaço de interface do usuário, incluindo HTML, CSS e JavaScript relacionados a esse pedaço específico.

O que é um componente?: Um componente no React é uma parte isolada e reutilizável da interface do usuário. Ele pode ser uma pequena parte, como um botão ou uma caixa de texto, ou uma parte maior, como um cabeçalho ou um formulário.

Criando componentes no React

Por que usar componentes?: Usar componentes torna o código mais organizado, fácil de entender e reutilizável. Em vez de ter uma grande quantidade de código misturado em um único arquivo, você pode dividir sua interface do usuário em componentes menores, cada um responsável por uma única funcionalidade ou visualização.

Criando componentes no React

```
//Pasta Botão - Aquivo index.js
import React from 'react';

function MeuBotao() {
  return (
    <button>Clique em mim</button>
  );
}

export default MeuBotao;
```

Criando componentes no React

Como usar um componente?: Depois de criar um componente, você pode usá-lo em qualquer lugar em sua aplicação React importando-o e renderizando-o como faria com qualquer outro elemento HTML. Por exemplo:

Criando componentes no React

```
import React from 'react';
import MeuBotao from './MeuBotao'; // Importa o componente MeuBotao

function App() {
  return (
    <div>
      <h1>Minha Aplicação</h1>
      <MeuBotao /> {/* Renderiza o componente MeuBotao */}
    </div>
  );
}

export default App;
```

Criando componentes no React - Exercício

Questão: Crie um componente React chamado Botao. Este componente deve representar um botão simples que exibe o texto "Clique Aqui". O componente deve ser uma função de seta e deve retornar um elemento <button> com o texto "Clique Aqui".

Instruções:

1. Crie um componente funcional chamado Botao.
2. Dentro do componente, retorne um elemento <button> com o texto "Clique Aqui".
3. Exporte o componente Botao para que ele possa ser usado em outros arquivos.

Importando um componente

Há duas formas de importar um componente. Importação padrão e importação de desestruturação.

Importar sem chaves (`import MeuComponente from './MeuComponente'`):

- Isso é chamado de importação padrão.
- Você usa isso quando exporta um único valor (como um componente) usando `export default`.
- Você pode nomear o componente importado como quiser.
- Exemplo:

Importando um componente

```
// MeuComponente.js
import React from 'react';

function MeuComponente() {
  return <div>Meu Componente</div>;
}

export default MeuComponente;

// OutroComponente.js
import MeuComponente from './MeuComponente';
```

Importando um componente

Importar com chaves (import { MeuComponente } from './MeuComponente');

- Isso é chamado de importação de desestruturação.
- Você usa isso quando exporta vários valores (como funções, constantes, etc.) usando export.
- Você deve usar o mesmo nome que foi usado para exportar.
- Exemplo:

Importando um componente

```
1 // MeuComponente.js
2 import React from 'react';
3
4 export function MeuComponente() {
5   return <div>Meu Componente</div>;
6 }
7
8 // OutroComponente.js
9 import { MeuComponente } from './MeuComponente';
```

Importando um componente

Então, em resumo:

- Use a importação padrão quando exportar usando `export default`.
- Use a importação de desestruturação quando exportar vários valores usando `export`.

Importando um componente - Exercício

Importe o componente botão que criamos anteriormente para a página de listas de usuários. Use a importação de desestruturação!

Propriedades no React

Passando dados para componentes: Você pode passar dados para componentes usando props (propriedades). As props são como atributos em elementos HTML e permitem que você passe dados de um componente pai para um componente filho. Por exemplo:

Propriedades no React

```
import React from 'react';

function Saudacao(props) {
  return <h1>Olá, {props.nome}!</h1>;
}

function App() {
  return (
    <div>
      <Saudacao nome="João" /* Passa o nome "João" para o componente Saudacao */>
      <Saudacao nome="Maria" /* Passa o nome "Maria" para o componente Saudacao */>
    </div>
  );
}

export default App;
```

Propriedades no React - Exercício

Questão: Crie um componente React chamado Cartao. Este componente deve exibir o conteúdo de um cartão, incluindo um título e uma descrição. O título e a descrição devem ser passados para o componente como props.

Instruções:

1. Crie um componente funcional chamado Cartao.
2. O componente deve receber duas props: titulo e descricao.
3. Dentro do componente, renderize um <div> que contenha:
 - Um elemento <h2> com o texto do título passado como prop titulo.
 - Um elemento <p> com o texto da descrição passada como prop descricao.
4. Exporte o componente Cartao para que ele possa ser usado em outros arquivos.

Propriedades no React - `props.children`

O `props.children` no React é uma propriedade especial que permite passar elementos filho para um componente. Isso significa que você pode envolver conteúdo dentro de um componente e acessá-lo através da propriedade `children` dentro do componente.

- 1. O que é o `props.children`?**: No React, `props.children` é uma propriedade especial que representa o conteúdo que está dentro de um componente na hierarquia de componentes.
- 2. Como funciona?**: Quando você usa um componente e coloca conteúdo dentro dele, esse conteúdo é passado automaticamente para o componente como `props.children`.

Propriedades no React - props.children

Exemplo: Por exemplo, suponha que você tenha um componente **Painel** que é usado para envolver outro conteúdo:

```
import React from 'react';

function Painel(props) {
  return (
    <div className="painel">
      {props.children}
    </div>
  );
}
```

Propriedades no React - props.children

Se você usar o componente Painel assim:

```
<Painel>
  <h2>Título do Painel</h2>
  <p>Conteúdo do Painel</p>
</Painel>
```

Então, dentro do componente Painel, `props.children` será igual a:

```
<div>
  <h2>Título do Painel</h2>
  <p>Conteúdo do Painel</p>
</div>
```

Propriedades no React - props.children

Se você usar o componente Painel assim:

```
<Painel>
  <h2>Título do Painel</h2>
  <p>Conteúdo do Painel</p>
</Painel>
```

Então, dentro do componente Painel, `props.children` será igual a:

```
<div>
  <h2>Título do Painel</h2>
  <p>Conteúdo do Painel</p>
</div>
```

Propriedades no React - props.children

Para que é usado?: O `props.children` é útil quando você deseja criar componentes mais genéricos que podem envolver qualquer tipo de conteúdo. Isso torna os componentes mais flexíveis e reutilizáveis.

Basicamente, o `props.children` permite que um componente receba e renderize conteúdo filho de forma dinâmica, o que é muito útil para criar componentes compostos e flexíveis no React.

Propriedades no React - props

Quando você passa props para um componente no React, você está basicamente passando um objeto contendo todas as propriedades que deseja compartilhar. Por exemplo:

Propriedades no React - props

```
// Componente Filho
function Filho(props) {
  return (
    <div>{/*O componente filho espera receber duas propriedades, nome e idade */}
      <p>Nome: {props.nome}</p>
      <p>Idade: {props.idade}</p>
    </div>
  );
}
```

```
// Componente Pai
function Pai() {
  return <Filho nome="João" idade={25} />;
}
```

Propriedades no React - props

No exemplo anterior, ao passar `nome="João"` e `idade={25}` para o componente `Filho`, você está passando duas `props`: `nome` e `idade`. Essas `props` são passadas como um objeto para o componente `Filho`, onde podem ser acessadas usando `props.nome` e `props.idade`.

Agora, sobre desestruturação de `props` utilizando chaves {}:

Propriedades no React - props

```
// Componente Filho
function Filho({ nome, idade }) {
  return (
    <div>
      <p>Nome: {nome}</p>
      <p>Idade: {idade}</p>
    </div>
  );
}
```

Propriedades no React - props

Agora, vamos atualizar o componente pai para desestruturar as props antes de passá-las para o componente filho:

```
// Componente Pai
function Pai() {
  const dados = { nome: "João", idade: 25 };
  return <Filho {...dados} />;
}
```

Propriedades no React - props

Neste exemplo, estamos desestruturando as props do componente filho (Filho) no componente pai (Pai) e passando-as como um único objeto (dados). Em seguida, estamos usando o spread operator (...dados) para passar essas props para o componente filho.

Propriedades no React - props

Ambos os códigos renderizarão o componente filho (Filho) com os mesmos dados (`nome: "João"` e `idade: 25`). No entanto, a desestruturação de `props` pode ser útil quando você tem muitas `props` para passar e deseja tornar o código do componente pai mais legível e conciso.

Props - Exercício

Questão: Crie um componente React chamado Painel. Este componente deve representar um painel simples com um título e um conteúdo. O título do painel deve ser passado como uma prop chamada `titulo`, e o conteúdo do painel deve ser passado entre as tags de abertura e fechamento do componente (`<Painel>` e `</Painel>`).

Instruções:

1. Crie um componente funcional chamado `Painel`.
2. O componente deve receber uma prop chamada `titulo`, que representa o título do painel.
3. Dentro do componente, renderize um `<div>` que contenha:
 - Um elemento `<h2>` com o texto do título passado como prop `titulo`.
 - O `props.children`, que representa o conteúdo passado entre as tags `<Painel>` e `</Painel>`.
4. Exporte o componente `Painel` para que ele possa ser usado em outros arquivos.

Revisando conceitos

O que é o Formato JSX?

O **JSX (JavaScript XML)** é uma extensão da sintaxe do JavaScript que permite escrever código HTML de forma direta dentro de componentes do React. Ele é usado para criar elementos de interface do usuário de forma declarativa e intuitiva no React. Em termos simples, o JSX é uma maneira de escrever HTML dentro do JavaScript, facilitando a criação de interfaces de usuário em aplicativos React. Ele ajuda a tornar o código mais legível e expressivo, pois combina a lógica JavaScript com a estrutura familiar do HTML.

Revisando conceitos

O que é a renderização no React?

A **renderização no React** é o processo de transformar componentes React em elementos visuais na tela do navegador.

O que é o DOM Virtual?

O **DOM Virtual** é uma representação em memória do DOM (Document Object Model) real. O React usa o DOM Virtual para comparar as diferenças entre o estado atual da interface do usuário e o estado anterior, para então atualizar somente o necessário no DOM real.

Revisando conceitos

Como funciona a Renderização no React?

Quando algo muda no estado ou nas props de um componente React, o React re-renderiza esse componente e seus componentes filhos. Em vez de atualizar diretamente o DOM real, o React compara o DOM Virtual com o DOM real e aplica apenas as alterações necessárias no DOM real.

Revisando conceitos

Como funciona a renderização condicional no React?

A **renderização condicional no React** é uma técnica que permite mostrar ou ocultar elementos na interface do usuário com base em condições específicas. É como se você dissesse ao React: "**Mostre este elemento se esta condição for verdadeira, senão, não mostre**".

Revisando conceitos

Como é feita a renderização de listas no React?

A **renderização de listas no React** é o processo de exibir elementos de uma matriz (ou qualquer estrutura de dados iterável) como uma lista na interface do usuário. Isso é feito usando o método `map()` para iterar sobre os elementos da matriz e criar elementos JSX para cada um deles.

Aprendemos que precisamos ter um array, usar o método `map()`, renderizar os elementos jsx dentro do método `map()`, e usar uma chave única!

Revisando conceitos

O que é um componente?

Um componente no React é uma parte reutilizável e isolada da interface do usuário que contém uma mistura de código JavaScript, HTML e CSS. Ele pode representar qualquer coisa, desde um botão simples até uma seção inteira da página.

Como exportar um componente?

Aprendemos duas formas de exportação: Colocar o export default nome do componente ao final do código, ou, ao criar a função, usar a estrutura export function nome do componente.

Revisando conceitos

Como importar um componente?

Basta usar a palavra `import nome do componente from "caminho para o componente"`

Existem duas formas de importação, quais são elas?

Importação padrão e importação desestruturada. A padrão é quando usamos o `export default` no final do código. A desestruturada é quando usamos o `export function`. A maneira de importar quando é importação desestruturada muda um pouco. O nome do componente fica entre chaves.

Revisando conceitos

O que são propriedades no React?

As propriedades (props) no React são como informações ou configurações que você pode passar para um componente. É como enviar um pacote de dados de um componente pai para um componente filho.

O que é o `props.children`?

O `props.children` no React é uma propriedade especial que permite passar conteúdo entre as tags de abertura e fechamento de um componente. É como uma área reservada dentro de um componente onde você pode colocar qualquer tipo de conteúdo - texto, outros componentes, elementos HTML, etc.

Colocando em prática

Exercício: Lista de Tarefas

Crie um componente React chamado `ListaDeTarefas`. Este componente deve renderizar uma lista de tarefas. Cada tarefa deve ser representada como um item de lista (``) com um checkbox para indicar se a tarefa foi concluída ou não. As tarefas devem ser passadas para o componente como uma prop chamada `tarefas`.

Colocando em prática

Instruções:

1. Crie um componente funcional chamado `ListaDeTarefas`.
2. O componente deve receber uma prop chamada `tarefas`, que é uma matriz de objetos representando as tarefas. Cada objeto deve ter as propriedades `id` (identificador único da tarefa), `texto` (descrição da tarefa) e `concluida` (indicador se a tarefa foi concluída).
3. Dentro do componente, renderize uma lista não ordenada (``) que contenha um item de lista (``) para cada tarefa na matriz `tarefas`.
4. Para cada item de lista, renderize um checkbox (`input` do tipo `checkbox`) e o texto da tarefa.
5. Adicione uma classe CSS chamada `concluida` aos itens de lista cuja propriedade `concluida` seja `true`.
6. Implemente a funcionalidade de marcar/desmarcar uma tarefa como concluída ao clicar no checkbox.

Colocando em prática

Instruções:

1. Crie um componente funcional chamado `ListaDeTarefas`.
2. O componente deve receber uma prop chamada `tarefas`, que é uma matriz de objetos representando as tarefas. Cada objeto deve ter as propriedades `id` (identificador único da tarefa), `texto` (descrição da tarefa) e `concluida` (indicador se a tarefa foi concluída).
3. Dentro do componente, renderize uma lista não ordenada (``) que contenha um item de lista (``) para cada tarefa na matriz `tarefas`.
4. Para cada item de lista, renderize um checkbox (`input` do tipo `checkbox`) e o texto da tarefa.
5. Adicione uma classe CSS chamada `concluida` aos itens de lista cuja propriedade `concluida` seja `true`.
6. Implemente a funcionalidade de marcar/desmarcar uma tarefa como concluída ao clicar no checkbox.

Colocando em prática

Exemplo de uso:

```
const tarefas = [
  { id: 1, texto: 'Fazer compras', concluida: false },
  { id: 2, texto: 'Estudar React', concluida: true },
  { id: 3, texto: 'Passear com o cachorro', concluida: false }
];

<ListaDeTarefas tarefas={tarefas} />
```