REACT: NAVEGANDO NO MAR DE COMPONENTES

JÚLIO CÉSAR CANDEIA SIMÃO DE LIMA

O que são Componentes React?

Componentes são os blocos fundamentais de qualquer aplicação React. Eles permitem dividir a interface do usuário em partes reutilizáveis, tornando o código mais organizado e fácil de manter.

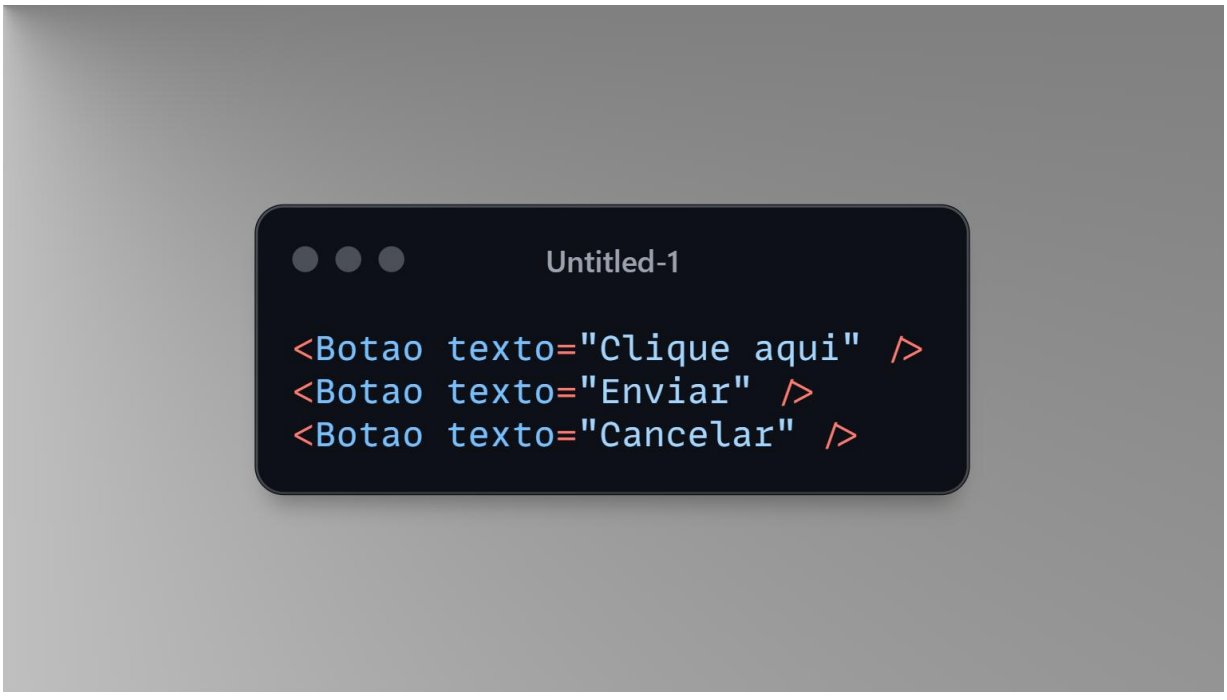
Exemplo simples: Um botão criado como um componente.

Untitled-1

```
function Botao(props) {
  return <button>{props.texto}</button>;
}

export default Botao;
```

Agora, você pode reutilizar o botão em várias partes da aplicação:



01

COMPONENTES FUNCIONAIS

Mais simples e modernos, escritos como funções JavaScript, ideais para a maioria dos casos.

Componentes Funcionais

Os componentes funcionais são escritos como funções JavaScript. São mais simples e recomendados na maioria dos casos.

Exemplo:

02

COMPONENTES DE CLASSE

Antes usados para funcionalidades avançadas, hoje são menos comuns devido aos hooks.

Componentes de Classe

Usados no passado para funcionalidades mais complexas, hoje eles são menos comuns devido ao advento dos hooks.

Exemplo:

Untitled-1

```
class Saudacao extends React.Component {
  render() {
    return <h1>Olá, {this.props.nome}!</h1>;
  }
}
```

03

PROPRIEDADES

Os props são usados para passar dados entre os componentes, tornando-os flexíveis e dinâmicos. São imutáveis e usados para customizar componentes.

Propriedades (Props): Comunicando Informações

Props permitem passar dados para os componentes, tornando-os dinâmicos.

Exemplo real: Um card de usuário.

Untitled-1

```
function UsuarioCard(props) {
  return (
    <div>
      <h2>{props.nome}</h2>
      <p>Email: {props.email}</p>
    </div>
  );
}

<UsuárioCard nome="João Silva" email="joao@email.com" />
```

04

ESTADO

O state permite que os componentes armazenem e atualizem informações dinamicamente, reagindo às interações do usuário, como cliques ou entradas de dados.

Estado (State): Tornando Componentes Interativos

O estado permite que os componentes "lembrem" informações e reajam às interações do usuário.

Exemplo: Contador que aumenta ao clicar.

Untitled-1

```jsx
import { useState } from 'react';

function Contador() {
  const [contagem, setContagem] = useState(0);

  return (
    <div>
      <p>Contagem: {contagem}</p>
      <button onClick={() => setContagem(contagem + 1)}>Aumentar</button>
    </div>
  );
}

export default Contador;
```

05

COMPOSIÇÃO

A composição combina pequenos componentes para criar estruturas mais complexas e reutilizáveis, promovendo um design modular e organizado.

Composição: Componentes Trabalhando Juntos

React incentiva a construção de interfaces com componentes menores que se unem para formar uma estrutura maior.

Exemplo real: Uma lista de tarefas.

Untitled-1

```jsx
function Tarefa({ texto }) {
  return <li>{texto}</li>;
}

function ListaDeTarefas() {
  const tarefas = ['Estudar React', 'Criar um projeto', 'Revisar código'];

  return (
    <ul>
      {tarefas.map((tarefa, index) => (
        <Tarefa key={index} texto={tarefa} />
      ))}
    </ul>
  );
}

export default ListaDeTarefas;
```

06

CICLO DE VIDA

Explica as fases pelos quais os componentes passam, como montagem, atualização e desmontagem. O hook useEffect facilita a integração dessas fases com ações, como buscar dados de uma API.

Ciclo de Vida do Componente

Os componentes seguem um ciclo de vida: criação, atualização e desmontagem. Com os hooks, como useEffect, é fácil gerenciar essas fases.

Exemplo: Buscar dados ao montar um componente.

Untitled-1

```jsx
import { useEffect, useState } from 'react';

function DadosApi() {
  const [dados, setDados] = useState([]);

  useEffect(() => {
    fetch('https://api.example.com/dados')
      .then((res) => res.json())
      .then((data) => setDados(data));
  }, []);

  return (
    <ul>
      {dados.map((item) => (
        <li key={item.id}>{item.nome}</li>
      ))}
    </ul>
  );
}

export default DadosApi;
```

07

Eventos: Tornando Componentes Mais Dinâmicos

Os eventos permitem capturar interações dos usuários, como cliques e digitação, e executar ações específicas. Com React, eventos são tratados de forma semelhante ao DOM, mas usando camelCase e funções.

Eventos: Tornando Componentes Mais Dinâmicos

Como lidar com eventos, como cliques, alterações em campos de entrada e submissões de formulários. Demonstradno como passar funções como propriedades para manipular comportamentos.

Exemplo.

Untitled-1

```
function BotaoInterativo() {
  const handleClick = () => alert("Botão clicado!");

  return <button onClick={handleClick}>Clique aqui</button>;
}
```

08

Estilizando Componentes

React oferece várias abordagens para estilizar componentes, desde CSS tradicional até soluções modernas como CSS-in-JS. Essas opções permitem criar interfaces visuais atraentes e consistentes.

Estilizando Componentes

Aborde maneiras de adicionar estilos aos componentes, como CSS tradicional, CSS-in-JS (ex.: Styled Components) ou bibliotecas como Material-UI.

Exemplo.

Untitled-1

```
function BotaoEstilizado() {
  const estilo = {
      backgroundColor: "blue",
      color: "white",
      padding: "10px" };

  return <button style={estilo}>Botão Estilizado</button>;
}
```

09

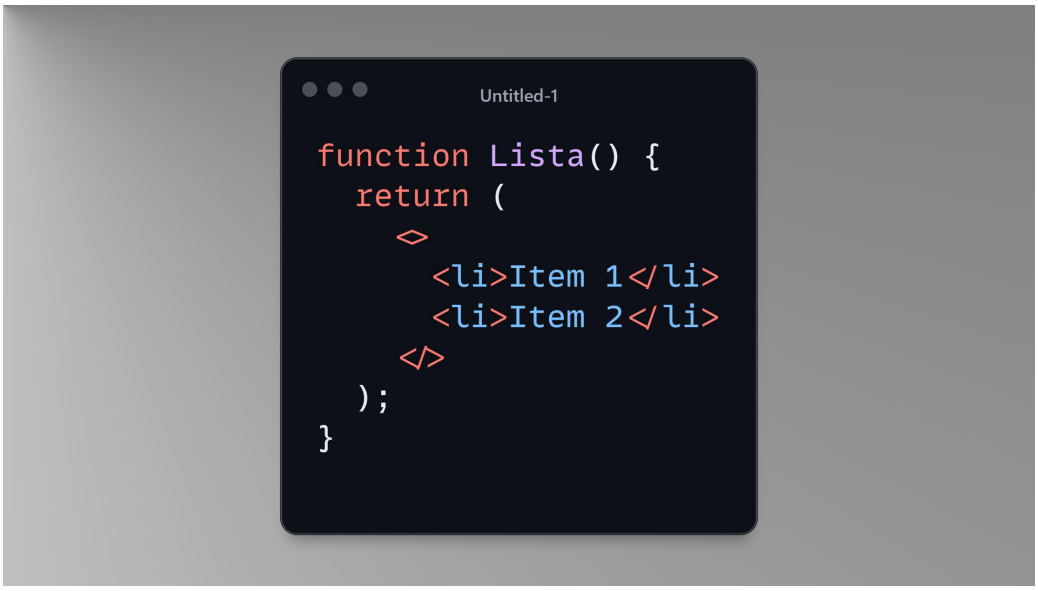
Fragments: Componentes Sem Elementos Externos

Os React.Fragment permitem agrupar elementos sem adicionar nós extras ao DOM, otimizando a estrutura e mantendo a semântica do HTML.

Fragments: Componentes Sem Elementos Externos

Introduza o uso de React.Fragment para evitar a criação de elementos DOM desnecessários ao agrupar múltiplos filhos.

Exemplo.



CONCLUSÃO

Componentes React são ferramentas incrivelmente poderosas para criar aplicações web dinâmicas e escaláveis. Comece com os conceitos simples, pratique e logo estará dominando o "leme" do seu próprio projeto React!

AGRADECIMENTOS

OBRIGADO POR LER ATÉ AQUI

Esse Ebook foi gerado por IA, e diagramdo e revisado por um humano

Este conteúdo foi gerado por fins de aprendizado, explorando ferramentas como chatGPT e WPS Office, o conteúdo na íntegra se encontra no meu GitHub.

GitHub

https://github.com/juliocesar710/ebookDIO