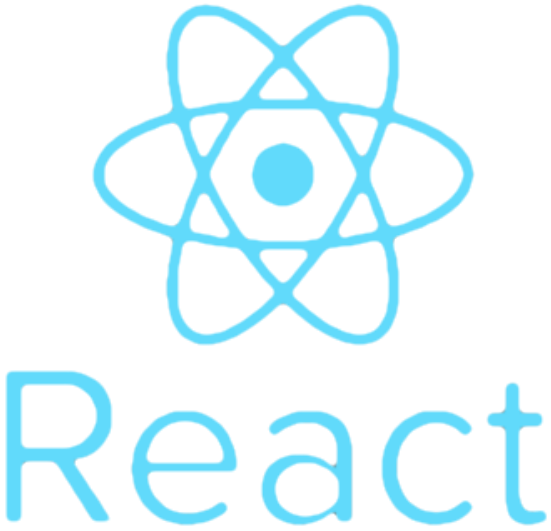
Índice

Assuntos: React

Abertura Alunos:
Apresentação pitch
Vitória Helena e Jorge Felipe.

React



O React é uma biblioteca modular(tem componentes reutilizado e compartilhado) de JavaScript de código aberto que foi desenvolvida pelo Facebook para construir interfaces de usuário (UI). Desenvolvida pelo Facebook, React permite a criação de componentes reutilizáveis que gerenciam seu próprio estado e podem ser compostos para construir interfaces de usuário complexas.

SPA

Single page application, ou aplicação de página única.
Que é carregada de forma dinâmica e atualizada conforme o cliente acessa a aplicação, ao invés de carregar uma nova página inteira a cada interação.
A página inteira é carregada de uma vez, o que permite o usuário acessar diferentes partes da página sem maiores carregamentos.

As principais características do React incluem:

Componentização: React permite a criação de componentes independentes e reutilizáveis, que podem ser facilmente combinados para formar interfaces de usuário complexas. Cada componente pode gerenciar seu próprio estado interno e pode ser facilmente reutilizado em diferentes partes do aplicativo.

Virtual DOM: React utiliza um conceito chamado Virtual DOM para aumentar a eficiência da renderização da interface do usuário. O Virtual DOM é uma representação em memória da estrutura de árvore de elementos da interface do usuário. Quando o estado de um componente muda, o React compara o Virtual DOM atual com uma versão anterior e atualiza apenas os elementos que foram alterados, minimizando assim o número de manipulações no DOM real.

JSX: JSX é uma extensão da sintaxe JavaScript que permite a definição de elementos de interface do usuário dentro do próprio código JavaScript. Isso torna a escrita de componentes React mais intuitiva e legível, ao misturar marcação HTML com lógica JavaScript.

Estado e Propriedades: React permite o gerenciamento eficiente do estado de um componente e a passagem de dados entre componentes pai e filho por meio de propriedades (props). O estado de um componente pode ser atualizado usando o método setState(), e as propriedades podem ser passadas de componentes pai para filhos para permitir a comunicação entre eles.

Reatividade: Quando o estado de um componente é alterado, o React automaticamente re-renderiza o componente e seus componentes filhos afetados para refletir essas mudanças na interface do usuário.

Em resumo, React é uma poderosa biblioteca JavaScript que simplifica o processo de construção de interfaces de usuário dinâmicas e reativas, facilitando a criação de aplicativos web e móveis modernos e eficientes.

Código Aberto : Isso significa que seu código-fonte é disponibilizado publicamente e pode ser acessado, visualizado e modificado por qualquer pessoa. Essa abertura permite que desenvolvedores de todo o mundo contribuam para o desenvolvimento do React, corrijam bugs, adicionem novos recursos e ofereçam suporte à comunidade de desenvolvimento.

O fato de o React ser de código aberto promove a transparência e a colaboração entre os desenvolvedores, ajudando a impulsionar a inovação e a evolução contínua da biblioteca. Além disso, permite que a comunidade de desenvolvedores compartilhe conhecimento, aprenda uns com os outros e crie soluções mais robustas e eficientes para problemas comuns de desenvolvimento de software.

Usamos o React para interface do usuário (facebook) que é atualizada frequentemente, atualiza muito mais rápido e com mais eficiência informações também, frequentemente usado com o REACT NATIVE para poder criar aplicações móveis nativas.

Aplicações Nativas

Aplicações nativas são aplicações nativas de sistemas como IOS e Android, etc…

Aplicações nativas referem-se a aplicativos de software desenvolvidos especificamente para uma plataforma ou sistema operacional específico, utilizando as linguagens de programação e ferramentas nativas daquela plataforma. Essas aplicações são otimizadas para funcionar em um ambiente específico, aproveitando as APIs (Interfaces de Programação de Aplicações) e os recursos oferecidos pela plataforma em questão.

Por exemplo:

Aplicações Nativas para Dispositivos Móveis:

iOS: Desenvolvidas em linguagens como Swift ou Objective-C, usando o ambiente de desenvolvimento Xcode.
Android: Desenvolvidas em Java ou Kotlin, usando o Android Studio como ambiente de desenvolvimento.

Aplicações Nativas para Desktop:

Windows: Desenvolvidas em linguagens como C# ou C++ usando a estrutura .NET ou outras ferramentas específicas do Windows.
macOS: Aplicações nativas são geralmente escritas em Swift, Objective-C ou, em alguns casos, em linguagens como C++.

Aplicações Web:

Em termos de aplicações web, quando se fala em nativo, pode-se referir a aplicações construídas utilizando tecnologias específicas da web, como HTML, CSS e JavaScript, sem depender de bibliotecas ou frameworks adicionais.
As aplicações nativas geralmente têm a vantagem de melhor desempenho, acesso total aos recursos do dispositivo e uma experiência do usuário mais integrada, uma vez que são desenvolvidas especificamente para o sistema operacional alvo. No entanto, elas podem demandar mais esforço de desenvolvimento, já que uma aplicação nativa para iOS, por exemplo, não será diretamente compatível com Android, e vice-versa. Para atender a ambas as plataformas, muitas vezes é necessário desenvolver versões separadas.

Componentes em React

Em React, os componentes são blocos de construção reutilizáveis para a construção de interfaces de usuário. Um componente React é uma parte autônoma e isolada da interface do usuário que pode ser criada, combinada e reutilizada em diferentes partes da aplicação.

- React ele usa uma linguagem js para criar seus componentes que são pedaços de código que eu vou estar representando na interface do código, cada componente tem um estado (exemplo: on ou off) cada estado é uma variável que eu vou estar armazenando algumas informações que mudam de um componentes para outro componente, os estados podem ser a cor de um botão, tamanho de imagem , dinâmica de link, etc.

DOM vs Virtual DOM

O DOM (Document Object Model) e o Virtual DOM são conceitos fundamentais em tecnologias web, especialmente em frameworks como React. Aqui está uma explicação sobre cada um deles e como se relacionam:

DOM (Document Object Model):

- O DOM é uma representação estruturada de um documento HTML (ou XML) que fornece uma interface para interagir com a estrutura e conteúdo do documento.
- Ele é representado como uma árvore de objetos, na qual cada nó representa um elemento (como tags HTML), atributos e texto.
- O DOM é dinâmico e pode ser modificado através de scripts JavaScript para alterar a estrutura, conteúdo e estilo de uma página web.
- Contras: Em estruturas mais complexas ele é lento e para resolver esse problema usa-se o Virtual DOM. **É a principal diferença entre o DOM e o Virtual DOM.**

Virtual DOM:

- O Virtual DOM é uma representação em memória da estrutura de árvore do DOM.
- Ele é uma cópia leve do DOM real, ou seja, é mais rápido e é mantido pelo React.
- Quando o estado de um componente React é atualizado, uma nova árvore Virtual DOM é criada e comparada com a árvore Virtual DOM existente.
- O React determina as diferenças entre as duas árvores e calcula a menor quantidade de manipulações de DOM necessárias para atualizar o DOM real.
- Depois de calcular as diferenças, o React atualiza apenas os nós do DOM que foram alterados, em vez de re-renderizar toda a página.

A principal razão para usar o Virtual DOM é otimizar o desempenho e a eficiência das atualizações do DOM. Manipular o DOM diretamente pode ser uma operação custosa em termos de desempenho, especialmente em aplicações web complexas. O Virtual DOM permite ao React minimizar o número de manipulações do DOM real, melhorando assim o desempenho da aplicação.

Em resumo, enquanto o DOM é a representação real da estrutura de uma página web, o Virtual DOM é uma camada intermediária mantida pelo React para otimizar as atualizações do DOM, garantindo um melhor desempenho em aplicações React.

- A atualização do DOM é feita primeiro no DOM Virtual e depois no DOM que está offline.
- E a memória do DOM Virtual está no servidor.

React - Implementação

Cookbooks de 2 a 4.

NPM Instalador de pacotes, tipo Maven, ele é tipo .exe de programas. Ele sempre baixa na web, é mais lento, mas não ocupa espaço na memória da máquina.

YARN é um gerenciador de pacotes para JavaScript, criado pelo Facebook em colaboração com outras empresas como Google, Exponent e Tilde. Ele foi desenvolvido para superar

algumas limitações do npm (Node Package Manager), o gerenciador de pacotes padrão do Node.js. a mesma coisa que o NPM só se diferencia pela máquina que está sendo usada. Yarn sempre vai baixar pela primeira vez na web e depois vai baixar na máquina (em cash), ele é mais rápido, porém ocupa espaço na memória.

NODE Node.js é um ambiente de execução JavaScript de código aberto, construído no motor JavaScript V8 do Chrome. Ele permite que os desenvolvedores usem JavaScript para escrever scripts do lado do servidor, o que antes era restrito principalmente ao lado do cliente em navegadores da web.

Aqui estão algumas características-chave do Node.js:

JavaScript no Servidor: Node.js permite que os desenvolvedores usem JavaScript tanto no lado do cliente quanto no lado do servidor. Isso significa que é possível escrever aplicativos completos, de ponta a ponta, usando apenas JavaScript.

Modelo de E/S Não-bloqueante: Node.js é conhecido por sua execução não bloqueante e orientada a eventos. Ele é particularmente eficiente para operações de I/O intensivas, como leitura e escrita em arquivos, acesso a bancos de dados e chamadas de API. Isso permite que o Node.js lide com um grande número de solicitações simultâneas com eficiência.

Módulos Nativos: Node.js possui um rico ecossistema de módulos e pacotes, tanto no NPM (Node Package Manager) quanto em módulos nativos. Os desenvolvedores podem usar esses módulos para adicionar funcionalidades extras aos seus aplicativos, como manipulação de arquivos, comunicação com bancos de dados, servidores HTTP, etc.

Rápido e Eficiente: Node.js é conhecido por seu desempenho rápido e eficiente. Ele utiliza o motor V8 do Chrome, que compila o JavaScript em código de máquina altamente otimizado. Além disso, sua arquitetura orientada a eventos e assíncrona o torna uma escolha popular para construir aplicativos escaláveis e de alto desempenho.

Comunidade Ativa: Node.js possui uma comunidade ativa e vibrante de desenvolvedores, que contribuem com bibliotecas, frameworks e ferramentas para melhorar e estender a funcionalidade do Node.js.

Em resumo, Node.js é uma poderosa plataforma que permite aos desenvolvedores construir rapidamente aplicativos escaláveis e de alto desempenho usando JavaScript no lado do servidor. Ele é amplamente utilizado em uma variedade de domínios, incluindo desenvolvimento web, IoT (Internet das Coisas), ferramentas de linha de comando, entre outros.

React - TypeScript

TypeScript é uma linguagem de programação que estende o JavaScript, adicionando tipagem estática opcional e outros recursos, como interfaces e enums.

Ele ajuda a detectar erros de código mais cedo no processo de desenvolvimento e oferece uma experiência de desenvolvimento mais robusta e escalável, especialmente para projetos grandes e complexos.

Arquivos TypeScript são de extensão **.tsx** .

Um arquivo .tsx é um arquivo de código-fonte TypeScript que contém código JSX usado para criar componentes React. Ele combina as vantagens do TypeScript, como verificação de tipos, com a facilidade de escrever código JSX para interfaces de usuário React.

React - Criação com Vite

- Crie uma pasta para seu projeto react.
- Abra essa pasta no VsCode, e abra o terminal.
- digite: **npm create vite@latest**
- Escolha o nome do projeto, se houverem espaços no nome, os substitua com ' **-** '.
- Escolha o tipo de projeto, nesse caso **React**.
- Escolha o tipo de escrita, nesse caso **TypeScript**.

(lembrando: **yarn** deve estar instalado no sistema)
Depois de finalizado, coloque:

- **cd *nome-projeto***
- **yarn**
- **yarn install react**
- **yarn run dev**

React - Componentes

Em React, os componentes são blocos de construção fundamentais usados para construir interfaces de usuário.

Eles são como peças de LEGO que podem ser reutilizadas e combinadas para criar interfaces complexas.

Cada componente pode conter seu próprio estado, propriedades e métodos, permitindo a criação de interfaces dinâmicas e interativas.

Em resumo, os componentes em React são unidades independentes e reutilizáveis de código que encapsulam a lógica e a apresentação de partes específicas da interface do usuário.

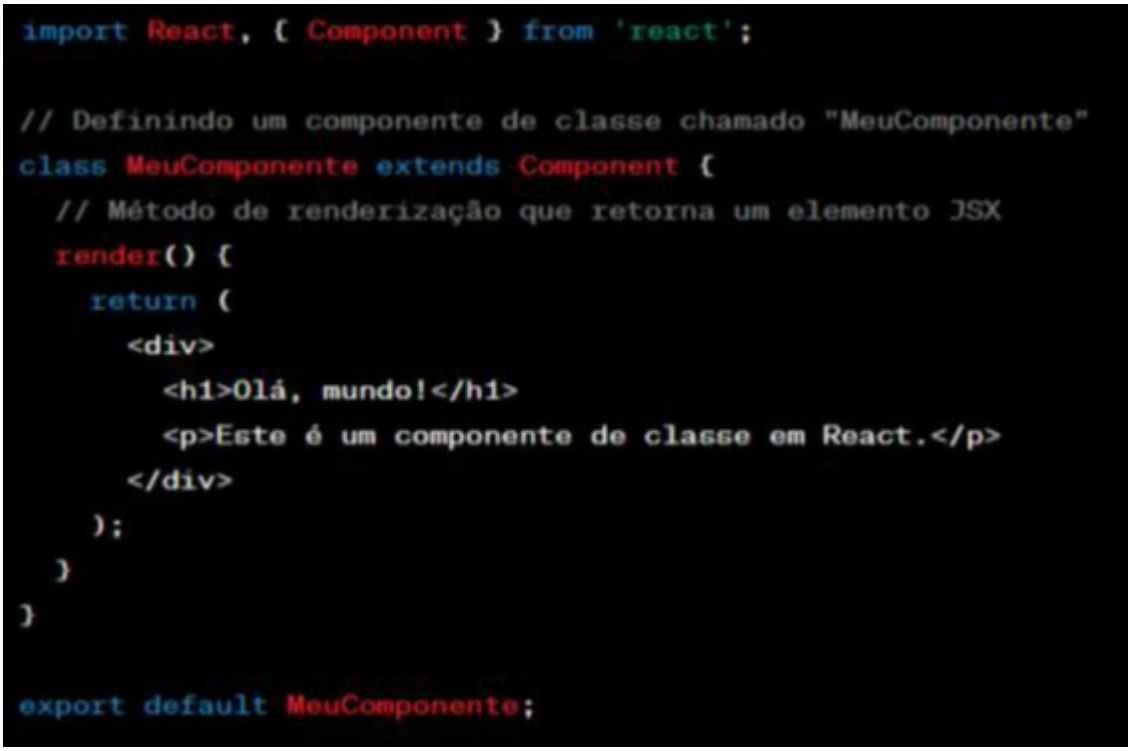
Existem dois tipos de componentes em react, os de função e os de classe, que como o nome já fala, são criados a partir de uma função, ou a partir de uma classe.

Exemplo de componente de função:

```jsx
import React from 'react';

// Definindo um componente de função chamado "MeuComponente"
function MeuComponente() {
  // Conteúdo do componente
  return (
    <div>
      <h1>Olá, mundo!</h1>
      <p>Este é um componente de função em React.</p>
    </div>
  );
}

export default MeuComponente;
```

```jsx
import React from 'react';

function MeuComponente() {
 return (
  <div>
   <h1>Bem-vindo ao meu aplicativo!</h1>
   <p>Espero que você esteja gostando.</p>
  </div>
 );
}

export default MeuComponente;
```

Exemplo componente de classe:



```jsx
import React, { Component } from 'react';

// Definindo um componente de classe chamado "MeuComponente"
class MeuComponente extends Component {
  // Método de renderização que retorna um elemento JSX
  render() {
    return (
      <div>
        <h1>Olá, mundo!</h1>
        <p>Este é um componente de classe em React.</p>
      </div>
    );
  }
}

export default MeuComponente;
```

```jsx
import React . (Component)  from 'react';

class MeuComponente extends React.Component {
 render() {
  return (
   <div>
    <h1>Bem-vindo ao meu aplicativo!</h1>
    <p>Espero que você esteja gostando.</p>
   </div>
  );
 }
}

export default MeuComponente;
```

É possível combinar componentes, para formar componentes maiores.

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

// Componente de classe que renderiza o componente de função
class App extends React.Component {
  render() {
    return (
      <div>
        <h2>Aplicativo React</h2>
        <MeuComponente />
      </div>
    );
  }
}

export default App;
```

```
import React from 'react';

class App extends React.Component {
 render() {
        return (
        <div>
        <h1>Meu Aplicativo</h1>
        <MeuComponente />
        </div>
        );
 }
}

export default App;
```

React - Fragments

Em React, um fragment é uma forma de agrupar elementos filhos sem adicionar um nó extra ao DOM. Ele permite que um componente retorne vários elementos filhos sem precisar envolvê-los em um elemento pai. Isso é especialmente útil em casos em que é necessário retornar uma lista de elementos sem um wrapper adicional, por exemplo.

Exemplo:

```
javascriptCopy codeimport React from 'react';

function ComponenteComFragment() {
 return (
        <>
        <p>Este é um parágrafo dentro do fragment.</p>
        <p>Este é outro parágrafo dentro do fragment.</p>
        </>
 );
}

export default ComponenteComFragment;
```

wrapper

Um "wrapper" (ou "embrulho", em português) é um conceito utilizado em programação para se referir a uma estrutura que envolve ou encapsula outro objeto ou componente. O objetivo principal de um wrapper é fornecer funcionalidades adicionais ou modificar o comportamento do objeto encapsulado, sem alterar sua própria estrutura.

React - Props

As props (abreviação de propriedades) em React são usadas para passar dados de um componente pai para um componente filho. Elas permitem que os componentes sejam configuráveis e reutilizáveis, tornando o código mais modular. As props são passadas para um componente como argumentos de função e são acessadas dentro do componente usando a sintaxe de ponto.

```
import React from 'react';

// Componente de função que recebe props
```

```
function MensagemBemVindo(props) {
 return <h1>Bem-vindo, {props.name}!</h1>;
}

// Componente principal que renderiza MensagemBemVindo e passa props
function App() {
 return <MensagemBemVindo name="João" />;
}

export default App;
```

Exemplo 2:

```
function Pai() {
 const dados = { nome: "João", idade: 30 };
 return <Filho dados={dados} />;
}

function Filho(props) {
 return (
        <div>
        <p>Nome: {props.dados.nome}</p>
        <p>Idade: {props.dados.idade}</p>
        </div>
 );
}
```

React - Hooks

Hooks são uma característica do React que permite adicionar funcionalidades de estado e ciclo de vida a componentes de função. Eles simplificam o compartilhamento de lógica entre componentes e promovem a reutilização de código.

Exemplo Hooks:

```
import React, { useState } from 'react';

function Counter() {
 const [count, setCount] = useState(0);
```

```
// Define uma variável de estado 'count' inicializada com o (zero)

 return (
        <div>
        <p>Você clicou  {count}  vezes</p>
        <button onClick={() => setCount(count + 1)}>
        Clique aqui
        </button>
        </div>
 );
}

export default Counter;
```

useState

O useState é um hook do React que permite que componentes funcionais tenham estado, ou seja, possam armazenar e atualizar dados ao longo do tempo. A estrutura básica do useState é a seguinte:

```
const [state, setState] = useState(initialState);
```

- **state** é a variável que armazena o estado atual do componente;
- **setState** é uma função que permite atualizar o valor de **state**;
- **useState** indica o estado a ser usado.
- **initialState** é o valor inicial de state.

Explicação Verônica:
É como um botão que ajuda a mudar e lembrar de informações importantes em um site. Se você quer que algo mude na página, você aperta esse botão para atualizar a informação. É como dizer ao React: "Ei, algo mudou, por favor, mostre na tela"

useEffect

O useEffect é um hook do React que permite que componentes funcionais tenham efeitos colaterais, ou seja, executem ações adicionais além da renderização da interface de usuário. Esses efeitos podem incluir atualizações de estado, chamadas de APIs, manipulação do DOM, entre outras.

```
useEffect(() => {
  // código que será executado
}, [dependencias]);
```

- A primeira argumento é uma função que será executada após a montagem do componente e também após cada atualização subsequente; (*//código que será executado*)
- O segundo argumento é um array opcional de dependências que, quando especificado, indica ao React quando a função deve ser executada. Se o valor de uma ou mais dependências mudar, a função será executada novamente. (*dependências*)

Explicação Verônica:
É como um assistente que ajuda a realizar tarefas especiais quando algo acontece em um site. Por exemplo, se você quiser buscar novas informações ou fazer algo extra quando a página carrega, você usa o useEffect. É como dizer ao site: "Quando algo importante acontecer, faça isso adicionalmente"

Renderização Condicional

A renderização condicional em React é uma técnica usada para renderizar diferentes elementos ou componentes com base em determinadas condições. Em outras palavras, é uma maneira de controlar quais elementos são exibidos na interface do usuário com base em alguma lógica definida pelo desenvolvedor.

A renderização condicional pode ser feita de várias maneiras em React, incluindo o uso de declarações "if" ou "switch" em conjunto com a lógica JavaScript ou expressões ternárias que avaliam uma condição e retornam um componente ou null.

Exemplo:

```
import React, { useState } from 'react';

function Home() {
  const [loggedIn, setLoggedIn] = useState(false);

  return (
        <div>
        {loggedIn ? (
        <h1>Bem-vindo de volta!</h1>
        ) : (
        <button onClick={() => setLoggedIn(true)}>Entrar</button>
```

```
        )}
        </div>
  );
}
export default Home;
```

Links e afins

CookBook, React, Introdução, MD01:
https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/05_react/01.md

CookBook, React, Ambiente de trabalho, MD02:
https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/05_react/02.md

CookBook, React, Criação pelo Vite, MD03:
https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/05_react/03.md

CookBook, React, Estilização CSS, MD04:
https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/05_react/04.md

React JS:
https://react.dev/

Yarn:
https://classic.yarnpkg.com/en/docs

Vite:
https://v2.vitejs.dev/guide/

Dúvidas