Índice

Assuntos: React - Props e Hooks.

Abertura Alunos:
Apresentação pitch
Fulano e Fulano.

JavaScript - Frameworks e Bibliotecas

Conjuntos de códigos pré escritos que fornece funcionalidades comuns e padronizadas para desenvolvimento de aplicações web front-end.

Incluem coisas como gerenciamento de estado, manipulação de eventos, animação, integração com apps, e outros.

As bibliotecas JavaScript são conjuntos de funções e métodos que são importados e usados em um projeto específico, permitindo que o desenvolvedor aproveite funcionalidades predefinidas em vez de criar tudo do 0.
Exemplos são: ReactJS, e Vue.Js.

Frameworks JavaScript são conjuntos de bibliotecas e ferramentas combinadas que fornecem uma estrutura completa para o desenvolvimento de aplicações web.

São mais abrangentes que as bibliotecas e incluem soluções pré definidas para problemas comuns, além de estruturas de arquitetura para organizar e gerenciar o código da aplicação.
Exemplos são: Angular.

Ambas bibliotecas e frameworks são usados no desenvolvimento de aplicações web front end, permitindo que os desenvolvedores criem rapidamente soluções escaláveis e confiáveis.

Vantagens de frameworks e bibliotecas front-end:

1. Produtividade
2. Padronização
3. Consistência
4. Compatibilidade
5. Comunidade

JavaScript - NodeJS

Node.js é uma plataforma JavaScript que permite aos desenvolvedores executar código do lado do servidor. Embora o Node.Js seja mais frequentemente associado ao desenvolvimento de backend, ele também pode ser usado em desenvolvimento frontend.

Uma das principais razões pelas quais os desenvolvedores front-end usam o Node.Js é para aproveitar o gerenciador de pacotes NPM (node package manager). NPM é um repositório online de bibliotecas de código aberto que os desenvolvedores podem instalar e usar em seus projetos de front-end.

Com NPM os desenvolvedores podem facilmente adicionar e gerenciar dependências de bibliotecas em seus projetos, o que ajuda a economizar tempo e aumentar a produtividade.

Outra vantagem é a capacidade de criar servidores de desenvolvimento local, que permite que os desenvolvedores vejam suas alterações em tempo real sem precisar carregar o código no servidor de produção.

React - Props

(Clique no título para ser direcionado as anotações anteriores de props)

Em React props é uma abreviação de properties (propriedades). Props são mecanismos de passagem de dados entre componentes.
São objetos que contém dados que são passados de um componente pai para um componente filho. Esses dados podem ser passados como atributos HTML, e em seguida, acessados dentro do componente filho do meio de suas propriedades.

Vantagens do uso de props:

1. Comunicação entre componentes
2. Reutilização de componentes
3. Manutenção simplificada
4. Flexibilidade
5. Legibilidade do código

Explicação Verônica:

São as mensagens que você envia para diferentes partes do seu site. Você usa props para contar a uma parte do site o que ela deve mostrar ou como deve se comportar. É como passar instruções especiais para diferentes partes de um quebra-cabeça

Exemplo:

```
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>
 );
}
```

Nesse exemplo, o componente pai está passando um objeto com as propriedades “nome” e “idade” para o componente filho por meio da propriedade dados. O componente filho acessa essas propriedades por meio do objeto props passado como argumento da função.

- No componente **Pai**, é definido um objeto **dados** com as propriedades **nome** e **idade**.
- O componente **Pai** renderiza o componente **Filho** e passa o objeto dados como propriedade **dados**.
- No componente **Filho**, as propriedades são acessadas usando **props.dados.nome** e **props.dados.idade** para exibir o nome e a idade recebidos do componente pai, respectivamente.
- Como resultado, o componente **Filho** renderiza um parágrafo com o nome e outro com a idade recebidos do componente **Pai**.

React - Hooks

(Clique no título para ser direcionado as anotações anteriores de hooks)
*Exemplos nas anotações anteriores

Em React um hook é uma função especial que permite que você use recursos do React, como estado, contexto, ciclo de vida do componente, e outros.

São uma maneira de compartilhar lógica entre componentes sem a necessidade de herança de classes. Existem vários hooks integrados ao React, como useState, useEffect, useContext, useMemo, useCallback, e outros, cada um desses hooks oferece um conjunto específico de recursos que podem ser usados em componentes funcionais.

Vantagens de uso de hooks:

1. Melhor reutilização do código
2. Mais legibilidade e simplicidade
3. Maior flexibilidade
4. Fácil migração de código
5. Melhor desempenho

Explicação Verônica:

São como truques especiais que ajudam a construir sites de forma mais fácil. Eles permitem que os sites lembrem de coisas importantes e façam truques legais, como mudar informações ou fazer coisas extras quando necessário.

React - 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.

Por exemplos, caso desejamos exibir uma mensagem diferente para usuários logados e não logados, podemos usar a renderização condicional para exibir o componente de login para o usuários não logados, e o componente de boas-vindas para usuários logados.

Podemos usar uma declaração if ou expressão ternária para verificar se o usuário está logado, e em seguida renderizar o componente apropriado com base nessa condição.

Exemplo simples:

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

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

 const handleLogin = () => {
        setLoggedIn(true);
 };

 let content;
 if (loggedIn) {
        content = <h1>Bem-vindo!</h1>;
 } else {
        content = <button onClick={handleLogin}>Login</button>;
 }

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

export default Login;
```

Explicação:

- O estado **loggedIn** controla se o usuário está logado ou não.
- Quando o usuário clica no botão "Login", a função **handleLogin** é chamada, e o estado **loggedIn** é atualizado para **true**.
- Dependendo do valor de **loggedIn**, o componente renderiza uma mensagem de boas-vindas (<h1>Bem-vindo!</h1>) se o usuário estiver logado, ou um botão de login (<button>Login</button>) caso contrário.

EXPLICAÇÃO VERONICA

Props: são as mensagens que você envia para diferentes partes do seu site. Você usa props para contar a uma parte do site o que ela deve mostrar ou como deve se comportar. É como passar instruções especiais para diferentes partes de um quebra-cabeça

Hooks: são como truques especiais que ajudam a construir sites de forma mais fácil. Eles permitem que os sites lembrem de coisas importantes e façam truques legais, como mudar informações ou fazer coisas extras quando necessário.

setState: 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: 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"

Naty, eu coloquei em cada local individual

Links e afins

Cookbook, React, Props, Hooks e Renderização Condicional, MD05:
https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/05_react/05.md

RoadMap, site de caminhos de estudos de programação:
https://roadmap.sh/

Indicação de estudo Veronica:
https://developer.mozilla.org/pt-BR/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started

---

Dúvidas