1. Explique a estrutura básica de um arquivo .vue. Quais são as principais seções e suas finalidades?
Um arquivo .vue é um **Single File Component** com três seções principais:

- <template>: define o **HTML** da interface.
- <script>: contém a **lógica JavaScript** do componente (dados, métodos, ciclo de vida).
- <style>: define o **CSS** para estilizar o componente.

Essas seções organizam a estrutura visual, funcional e estética do componente em um só lugar.

2. Defina o que são props no Vue.js e como elas são usadas para comunicação entre componentes.
Props são propriedades usadas para **passar dados de um componente pai para um componente filho**.
Elas permitem a **comunicação unidirecional**, ou seja, o pai envia valores ao filho, que os recebe como variáveis somente leitura.

3. Explique a diferença entre props e eventos personalizados na comunicação entre componentes no Vue.js. Quando usar cada um deles?

- **Props** são usadas para **enviar dados do componente pai para o filho**.
- **Eventos personalizados** ($emit) são usados para **enviar informações do filho de volta ao pai**.

Quando usar:

- Use **props** quando o pai quiser **fornecer dados** ao filho.
- Use **eventos** quando o filho quiser **notificar** o pai sobre uma ação (ex: clique ou envio de formulário).

4. O que são slots no Vue.js e qual a sua principal utilidade? Dê um exemplo de como eles podem ser usados para personalizar componentes.
Slots são áreas reservadas dentro de um componente para inserir **conteúdo personalizado**.

Eles permitem que o componente pai envie **HTML dinâmico** para ser exibido dentro do componente filho.

```
<!-- ComponenteFilho.vue -->

<template>

 <div class="caixa">

  <slot></slot>

 </div>

</template>
<!-- ComponentePai.vue -->

<ComponenteFilho>

 <p>Texto inserido no slot!</p>

</ComponenteFilho>
```

5. Descreva o conceito de mixins no Vue.js. Quais são os benefícios e os possíveis problemas de usá-los em projetos grandes?

Mixins são objetos reutilizáveis que contêm opções de componentes (como métodos, dados e hooks) que podem ser compartilhados entre vários componentes Vue.

Benefícios:

- Reutilização de código comum em vários componentes.
- Organização melhor de funcionalidades repetidas.

Possíveis problemas:

- **Conflito de nomes** entre propriedades ou métodos.
- **Dificuldade de rastrear** a origem de funcionalidades em projetos grandes, o que pode prejudicar a manutenção.

6. Explique a importância de utilizar o atributo v-bind:key ao trabalhar com listas dinâmicas em um componente.

O v-bind:key é essencial para **identificar cada item de forma única** em listas renderizadas com v-for.

Ele ajuda o Vue a **rastrear e atualizar os elementos corretamente**, evitando

recriações desnecessárias e melhorando a performance e a consistência da interface.

7. Quais são as vantagens de organizar um projeto Vue.js utilizando pastas como components/, views/, router/ e store/?
Essa organização traz **clareza, escalabilidade e manutenibilidade** ao projeto:

- **components/**: armazena componentes reutilizáveis.
- **views/**: contém páginas completas (geralmente ligadas às rotas).
- **router/**: gerencia as rotas da aplicação.
- **store/**: centraliza o estado global (Vuex ou Pinia).

8. O que acontece se dois componentes diferentes utilizarem o mesmo mixin? Como o Vue trata possíveis conflitos de métodos ou dados?
Se dois componentes diferentes usarem o mesmo **mixin**, o conteúdo do mixin será **mesclado** com o do componente.

Se houver **conflitos** (ex: métodos ou dados com o mesmo nome), o Vue **prioriza o que está no componente**, sobrescrevendo o que veio do mixin.
Isso evita que o mixin quebre o componente, mas pode causar **comportamentos inesperados** se não for bem documentado.

9. Qual a diferença entre usar um slot simples (<slot></slot>) e um slot nomeado (<slot name="..."></slot>) em um componente Vue?

- **Slot simples**: insere **qualquer conteúdo padrão** enviado pelo componente pai.
- **Slot nomeado**: permite **vários espaços personalizados**, cada um identificado por um name.

10. Descreva como o Vue.js facilita a comunicação de dados entre componentes pais e filhos. Por que isso é importante para aplicações modulares?

O Vue.js facilita a comunicação entre componentes pai e filho usando:

Props para enviar dados do pai para o filho.

Eventos personalizados ($emit) para o filho informar ações ao pai.

Essa estrutura é importante porque mantém a aplicação organizada, previsível e modular, permitindo a reutilização de componentes e facilitando a manutenção em projetos maiores.