1. O que é o Pinia e quais são suas principais vantagens em relação ao Vuex?

Pinia é uma biblioteca oficial para gerenciamento de estado no Vue.js, considerada sucessora do Vuex.

Principais vantagens sobre o Vuex:

- API mais simples e intuitiva.
- Melhor suporte a TypeScript.
- Menos boilerplate e configuração.
- Suporta composição e modularização mais fácil.
- Melhor desempenho e integração com Vue 3.

2. Explique os conceitos de state, actions, mutations e getters no Pinia. Como cada um deles é utilizado?

- **State:** armazena os dados reativos do estado da aplicação.
- **Actions:** funções que alteram o estado e podem conter lógica assíncrona.
- **Mutations:** no Pinia, não existem explicitamente; as mudanças de estado ocorrem diretamente dentro das actions.
- **Getters:** funções que retornam dados derivados do estado, similares a propriedades calculadas.

No Pinia, as **actions** substituem mutations, simplificando a manipulação do estado.

3. Qual a importância da modularização do Pinia? Como ela pode ser implementada em um projeto Vue.js?

A modularização no Pinia permite dividir o estado da aplicação em **múltiplas stores independentes**, facilitando organização, manutenção e escalabilidade.

Como implementar:

- Criar arquivos separados para cada store (ex: userStore.js, productStore.js).
- Registrar cada store com defineStore e importá-las onde necessário.

- Isso permite carregar só o estado relevante em cada parte da aplicação, melhorando desempenho e clareza do código.

4. Por que é importante gerenciar métodos assíncronos no Pinia? Dê um exemplo de como isso pode ser feito.

Gerenciar métodos assíncronos no Pinia é importante para lidar com operações como chamadas a APIs, onde o estado precisa ser atualizado após a resposta.

No Pinia, métodos assíncronos são implementados dentro das **actions**, que podem usar async/await para controlar o fluxo e atualizar o estado quando a operação terminar.

```
actions: {
 async fetchUsers() {
  const response = await fetch('https://api.example.com/users');
  this.users = await response.json();
 }
}
```

5. O que é o Vue DevTools e como ele pode ser utilizado para monitorar o estado global do Pinia?

O **Vue DevTools** é uma extensão de navegador que permite inspecionar e depurar aplicações Vue.js.

Para o Pinia, ele mostra:

- O estado atual das stores.
- Mudanças no estado em tempo real.
- Histórico de ações executadas.

Isso ajuda desenvolvedores a monitorar, entender e corrigir o fluxo de dados na aplicação de forma visual e interativa.

6. Explique os princípios de Clean Code: DRY, KISS e SOLID. Como aplicá-los no desenvolvimento Vue.js?

- **DRY (Don't Repeat Yourself):** evite duplicação de código. Em Vue.js, reutilize componentes e funções para manter o código limpo.
- **KISS (Keep It Simple, Stupid):** mantenha o código simples e direto. Evite complexidade desnecessária nos componentes e lógica.
- **SOLID:** conjunto de princípios para design de código (responsabilidade única, aberto/fechado, substituição de Liskov, segregação de interface, inversão de dependência). No Vue.js, aplique separando responsabilidades em componentes e serviços, usando props e eventos claros.

Seguir esses princípios melhora a manutenção, legibilidade e escalabilidade do projeto.

7. Como deve ser a estrutura ideal de um projeto Vue.js grande e escalável? Liste as principais pastas e sua finalidade.

- **src/components/**: componentes reutilizáveis pequenos e médios.
- **src/views/**: componentes de página ou telas completas.
- **src/router/**: configuração das rotas da aplicação.
- **src/store/**: gerenciamento de estado (Pinia ou Vuex).
- **src/assets/**: arquivos estáticos como imagens, fontes e estilos.
- **src/plugins/**: plugins e configurações externas.
- **src/utils/**: funções utilitárias e helpers.
- **src/services/**: chamadas a APIs e lógica de comunicação externa.

8. O que são ESLint e Prettier? Como essas ferramentas ajudam no desenvolvimento Vue.js?

- **ESLint:** ferramenta de análise estática que detecta problemas e padrões ruins no código, garantindo qualidade e boas práticas.

- **Prettier:** formatador automático que padroniza o estilo do código (indentação, espaços, quebras de linha).

No desenvolvimento Vue.js, elas ajudam a manter o código consistente, legível e livre de erros comuns, facilitando a colaboração em equipe e a manutenção.

9. Por que é importante padronizar código em projetos colaborativos? Como ferramentas como Prettier auxiliam nisso?

Padronizar código é importante para que todos os desenvolvedores sigam o mesmo estilo, facilitando leitura, revisão e manutenção do projeto.

O **Prettier** automatiza essa padronização, formatando o código automaticamente conforme regras definidas, evitando discussões sobre estilo e reduzindo erros causados por formatação inconsistente.

10. O que é a Composition API e como ela se relaciona com o Pinia no Vue.js?

A **Composition API** é uma forma de organizar a lógica dos componentes Vue.js usando funções reutilizáveis, que melhora a modularidade e legibilidade.

O **Pinia** integra-se bem com a Composition API, permitindo definir stores com defineStore usando funções reativas e hooks, facilitando o gerenciamento de estado de forma mais flexível e moderna.