===========================================================

***Questões***

===========================================================

**1) Descreva e explique todo Ciclo de Vida do VueJs.**

O Vue.js é um framework JavaScript progressivo amplamente utilizado para a construção de interfaces de usuário (UI) interativas e reativas.

O ciclo de vida do Vue.js é uma parte fundamental do framework e ajuda os desenvolvedores a entender o momento em que os ganchos (ou "hooks") personalizados podem ser utilizados para controlar o comportamento e a lógica de componentes Vue.

O ciclo de vida do Vue.js é dividido em várias etapas, e aqui estão elas com explicações:

1. ***Criação (Creation):***

beforeCreate: Neste ponto, o componente está sendo inicializado, mas as opções do componente ainda não estão acessíveis. Este é o momento de configurar dados iniciais e realizar ações antes da criação do componente.

created: O componente foi criado com sucesso. Neste ponto, os dados e métodos do componente estão acessíveis. É um bom lugar para fazer solicitações de API ou configurações iniciais.

1. ***Montagem (Mounting):***

beforeMount: O template do componente está compilado, mas ainda não está associado ao DOM. Este é o ponto para acessar o DOM antes da montagem.

mounted: O componente foi montado no DOM com sucesso. Neste momento, você pode acessar os elementos do DOM e realizar operações relacionadas à interface do usuário.

1. ***Atualização (Updating):***

beforeUpdate: Este é acionado antes que os dados do componente sejam atualizados, mas o DOM ainda não foi renderizado. Você pode realizar operações de pré-atualização, como comparação de dados.

updated: Após a atualização do DOM, este gancho é acionado. É um ótimo lugar para executar ações que dependem do DOM ou que precisam ser realizadas após uma atualização de dados.

1. ***Destruição (Destruction):***

beforeDestroy: Neste ponto, o componente está prestes a ser destruído, mas ainda é completamente funcional. É um momento para limpar event listeners ou cancelar assinaturas.

destroyed: O componente foi completamente destruído. Nenhuma referência ao componente deve existir neste ponto.

Além dessas etapas principais, o Vue.js também oferece ganchos específicos para erros, como "errorCaptured", que permitem lidar com erros de componentes filhos.

O Ciclo de Vida do Vue.js é fundamental para entender quando e onde implementar lógica específica em um componente Vue.

Cada etapa fornece um ponto de entrada para personalizar o comportamento do componente, como configurar dados, interagir com o DOM, lidar com atualizações ou executar tarefas de limpeza na destruição.

Isso permite o desenvolvimento de aplicativos Vue.js altamente controláveis e adaptáveis, mantendo o código organizado e fácil de gerenciar.

**2) Quais os passos para utilizar o VueJs em aplicações WEB?**

1. ***Configurar um Ambiente de Desenvolvimento:***

Certifique-se de ter um ambiente de desenvolvimento JavaScript configurado, incluindo Node.js e npm (Node Package Manager) . Você pode baixar e instalar o Node.js no site oficial.

1. ***Criar um Projeto Vue:***

Use o Vue CLI (Command Line Interface) para criar um novo projeto Vue. O Vue CLI é uma ferramenta oficial para criar projetos Vue.js com configurações pré-definidas. Você pode instalá-lo globalmente usando o npm:

npm install -g @vue/cli

Depois de instalado, crie um novo projeto Vue usando o comando vue create :

vue create meu-projeto

1. ***Configurar o Projeto:***

O Vue CLI oferece opções para personalizar a configuração do projeto, como a escolha de um modelo de projeto, configurações de linting e seleção de recursos. Escolha as configurações que se adequam ao seu projeto.

1. ***Desenvolver Componentes Vue:***

O Vue.js é baseado em componentes. Crie componentes Vue para gerenciar a interface do usuário e a lógica do aplicativo. Você pode criar componentes usando arquivos .vue que contêm modelos HTML, scripts JavaScript e estilos CSS .

1. ***Integrar Componentes em Páginas:***

Integre os componentes Vue em páginas HTML. Use as diretivas Vue, como v-bind e v-on, para conectar dados e métodos do componente à interface do usuário.

1. ***Gerenciar Dados com Vue:***

Use a instância Vue para gerenciar dados reativos. Você pode definir dados, métodos , computeds e watchers para controlar o comportamento do componente.

1. ***Roteamento (Opcional):***

Se a sua aplicação requer navegação entre diferentes páginas, você pode integrar uma biblioteca de roteamento, como o Vue Router, para gerenciar as rotas da sua aplicação.

1. ***Estado Global (Opcional):***

Para compartilhar dados e estados entre componentes, você pode usar o VueX, que é uma biblioteca de gerenciamento de estado para aplicações Vue.js .

1. ***Estilo (Opcional):***

Estilize seus componentes e páginas usando CSS ou uma pré-processador CSS como Sass ou Less, dependendo das configurações do seu projeto.

1. ***Build e Deploy:***

Use comandos do Vue CLI para compilar e empacotar seu projeto para produção. Você pode usar npm run build para criar os arquivos otimizados para produção.

1. ***Implantação:***

Implante sua aplicação em um servidor web ou em uma plataforma de hospedagem, como Netlify, Vercel, ou outras opções disponíveis.

1. ***Testes (Opcional):***

Implementar testes unitários e de integração para garantir a qualidade e o bom funcionamento da sua aplicação.

Esses são os passos gerais para utilizar o Vue.js em aplicações web. O Vue CLI simplifica o processo de configuração e desenvolvimento, permitindo que você se concentre na criação de recursos e na lógica do aplicativo.

**3) Explique o que é two way data binding e como funciona no VueJs.**

O Two-Way Data Binding é um conceito que permite que as alterações nos dados do modelo (ou estado) de um aplicativo sejam refletidas automaticamente na interface do usuário (UI) e vice-versa.

No contexto do Vue.js , o Two-Way Data Binding é uma das características fundamentais que tornam o desenvolvimento de interfaces interativas e reativas mais fácil.

No Vue.js, o Two-Way Data Binding é alcançado principalmente usando a diretiva v-model .

1. **Binding de Dados de um Componente para a UI:**

Quando você usa a diretiva v-model em um elemento de formulário, como um campo de texto (<input>), um elemento <textarea> ou um elemento <select>, você está criando um vínculo de dados bidirecional entre o valor do modelo de dados e o valor do campo de formulário na UI .

Por exemplo, ao usar v-model="message" em um campo de texto, o valor de message no modelo de dados será automaticamente refletido no campo de texto na UI .

1. **Atualizações de Dados de UI para o Modelo:**

Quando o usuário interage com a UI, como digitando em um campo de texto, as alterações são refletidas automaticamente no modelo de dados subjacente. Isso significa que, quando o usuário digita algo no campo de texto, o valor da variável message no modelo também é atualizado automaticamente.

1. **Atualizações de Dados de Modelo para a UI:**

Da mesma forma, se você atualizar o valor da variável message no modelo por meio de código, a UI será automaticamente atualizada para refletir essa mudança. Portanto, se você definir message como uma nova string no JavaScript, a alteração será refletida no campo de texto sem a necessidade de manipulação adicional do DOM.

O Two-Way Data Binding no Vue.js simplifica o desenvolvimento de interfaces de usuário reativas, eliminando a necessidade de atualizações manuais do DOM. Isso torna a criação de aplicativos Vue.js mais eficiente e produtiva, pois os desenvolvedores podem se concentrar na lógica do aplicativo e na experiência do usuário em vez de lidar com manipulações complexas do DOM.

**4) Qual a diferença de Mustache e Diretiva?**

Mustache e Diretivas são dois conceitos frequentemente usados no desenvolvimento front-end, especialmente em frameworks e bibliotecas JavaScript , como o Vue.js . Eles têm finalidades diferentes e são usados em contextos diferentes:

* **Mustache:**

O Mustache é uma linguagem de modelo que permite a inserção de variáveis e expressões em um modelo de texto. Ela usa duplas chaves ({{}}) para envolver variáveis ou expressões, que são substituídas pelos valores reais quando o modelo é renderizado. O Mustache é uma forma simples de interpolação de dados.

**Exemplo Mustache:**

<p>Olá, {{nome}}</p>

O Mustache não executa lógica condicional, loops ou qualquer outra lógica complexa no modelo. Ele é puramente para substituição de variáveis no texto.

* **Diretivas:**

As diretivas são recursos ou instruções especiais fornecidos por frameworks como Vue.js. Elas são usadas para adicionar funcionalidades dinâmicas a elementos HTML. Diretivas são prefixadas com um identificador especial, como v- no caso do Vue.js , seguido do nome da diretiva.

**Exemplo de diretiva Vue.js:**

<button v-on:click="mostrarMensagem">Clique-me</button>

Neste exemplo, v-on:click é uma diretiva que adiciona um evento de clique ao botão, fazendo com que a função mostrarMensagem seja executada quando o botão é clicado. As diretivas podem fazer coisas como manipular eventos, controlar classes e estilos, realizar loops, criar estruturas condicionais e muito mais.

Em resumo, a diferença fundamental é que o Mustache é usado para interpolação de dados em modelos, enquanto as diretivas são usadas para adicionar comportamento dinâmico e funcionalidades específicas a elementos HTML .

Diretivas são mais poderosas e versáteis, permitindo que você crie aplicativos interativos e reativos, enquanto o Mustache é simplesmente uma maneira de substituir valores de variáveis em texto estático.

**5) Assinale a alternativa que apresenta o que é o Vue.js e suas principais características.**

É um framework JavaScript para construir interfaces de usuário e suas principais características incluem data binding bidirecional, componentização e gerenciamento eficiente de estado.

É uma linguagem de programação orientada a objetos com uma sintaxe similar ao JavaScript e é usada para construir interfaces de usuário.

É um banco de dados não relacional amplamente utilizado para armazenar dados de aplicativos web.

É um framework CSS para estilizar páginas web de forma eficiente.

Vue.js é, de fato, um framework JavaScript para construir interfaces de usuário.

Suas Principais Características incluem data binding bidirecional , que permite atualizações automáticas entre o modelo e a interface do usuário, componentização, que facilita a construção de aplicativos com componentes reutilizáveis, e o gerenciamento eficiente de estado, o que é facilitado pelo uso do VueX para o controle de estado em grande escala.

**6) O que é um componente em Vue.js?**

Um componente é uma função JavaScript que retorna uma interface de usuário.

Um componente é um objeto JavaScript que armazena dados e funções relacionadas.

Um componente é uma instância de Vue que pode ser reutilizada em seu aplicativo.

Um componente é uma variável que armazena informações sobre a aplicação.

Em Vue.js , um componente é uma instância de Vue que encapsula dados, funcionalidades e uma interface de usuário relacionados. Um componente pode ser reutilizado em diferentes partes de seu aplicativo, tornando-o uma unidade modular e reutilizável.

**7) Como pode ser incorporado condicional em um modelo Vue.js?**

Utilizando a diretiva v-if.

Utilizando a diretiva v-for.

Utilizando a diretiva v-else.

Utilizando a diretiva v-switch.

No Vue.js , você pode incorporar condicionais em um modelo usando a diretiva v-if . Ela permite que você condicionalmente renderize ou não um elemento com base em uma expressão.

**8) Como podemos utilizar eventos em Vue.js?**

Utilizando a diretiva v-on.

Utilizando a diretiva v-emit.

Utilizando a diretiva v-bind.

Utilizando a diretiva v-event.

Para utilizar eventos em Vue.js, você pode usar a diretiva v-on. Ela permite que você associe manipuladores de eventos a elementos na interface do usuário.

A diretiva v-emit não existe no Vue.js ; em vez disso, você pode usar o método $emit para emitir eventos personalizados.

**9) Qual é a principal vantagem de utilizar Vue.js?**

Facilita a manipulação do DOM de forma direta.

Torna desnecessário o uso de HTML e CSS.

Permite a criação de interfaces de usuário reativas e eficientes.

Simplifica a implementação de lógica de servidor.

A principal vantagem de utilizar Vue.js é que ele permite a criação de interfaces de usuário reativas e eficientes. Vue.js fornece recursos de data binding bidirecional , componentização e uma abordagem reativa que torna a atualização da interface do usuário em resposta a alterações nos dados (modelo) muito eficiente e simplificada.

**10) O que é a diretiva v-for em Vue.js e para que ela é utilizada?**

A diretiva v-for é utilizada para criar loops em modelos Vue.js e é especialmente útil para renderizar uma lista de itens com base em um array.

A diretiva v-for é utilizada para criar funções assíncronas em Vue.js e é especialmente útil para operações de I/O.

A diretiva v-for é utilizada para criar condicionais em Vue.js e é especialmente útil para controlar a visibilidade de elementos no DOM.

A diretiva v-for é utilizada para criar animações em Vue.js e é especialmente útil para tornar as transições de página mais suaves.

A diretiva v-for é usada para criar loops em Vue.js , permitindo que você renderize uma lista de itens com base em um array ou outro tipo de estrutura de dados iterável. Ela é frequentemente usada para renderizar repetidamente elementos com base em uma matriz de dados.

Ela não é usada para criar funções assíncronas, condicionais ou animações, mas sim para iteração e renderização de listas de elementos.