## ****vue第三周考试试题****

## ****1、v-show 与 v-if 有什么区别？（8）****

v-if 是真正的条件渲染，因为它会确保在切换过程中条件块内的事件监听器和子组件适当地被销毁和重建；也是惰性的：如果在初始渲染时条件为假，则什么也不做——直到条件第一次变为真时，才会开始渲染条件块。

v-show 就简单得多——不管初始条件是什么，元素总是会被渲染，并且只是简单地基于 CSS 的 “display” 属性进行切换。

所以，v-if 适用于在运行时很少改变条件，不需要频繁切换条件的场景；v-show 则适用于需要非常频繁切换条件的场景。

## ****2、Class 与 Style 如何动态绑定？（8）****

Class 可以通过对象语法和数组语法进行动态绑定：

对象语法：

<div v-bind:class="{ active: isActive, 'text-danger': hasError }"></div>

数组语法：

<div v-bind:class="[isActive ? activeClass : '', errorClass]"></div>

Style 也可以通过对象语法和数组语法进行动态绑定：

对象语法：

<div v-bind:style="{ color: activeColor, fontSize: fontSize + 'px' }"></div>

数组语法：

<div v-bind:style="[styleColor, styleSize]"></div>

## ****3、computed 和 watch 的区别和运用的场景？（6）****

computed： 是计算属性，依赖其它属性值，并且 computed 的值有缓存，只有它依赖的属性值发生改变，下一次获取 computed 的值时才会重新计算 computed 的值；

watch： 更多的是「观察」的作用，类似于某些数据的监听回调 ，每当监听的数据变化时都会执行回调进行后续操作；

运用场景：

当我们需要进行数值计算，并且依赖于其它数据时，应该使用 computed，因为可以利用 computed 的缓存特性，避免每次获取值时，都要重新计算；

当我们需要在数据变化时执行异步或开销较大的操作时，应该使用 watch，使用 watch 选项允许我们执行异步操作 ( 访问一个 API )，限制我们执行该操作的频率，并在我们得到最终结果前，设置中间状态。这些都是计算属性无法做到的。

## ****4、Vue 的父组件和子组件生命周期钩子函数执行顺序？（8）****

Vue 的父组件和子组件生命周期钩子函数执行顺序可以归类为以下 4 部分：

加载渲染过程

父 beforeCreate -> 父 created -> 父 beforeMount -> 子 beforeCreate -> 子 created -> 子 beforeMount -> 子 mounted -> 父 mounted

子组件更新过程

父 beforeUpdate -> 子 beforeUpdate -> 子 updated -> 父 updated

父组件更新过程

父 beforeUpdate -> 父 updated

销毁过程

父 beforeDestroy -> 子 beforeDestroy -> 子 destroyed -> 父 destroyed

## ****5、Vue 组件间通信有哪几种方式？（10）****

Vue 组件间通信是面试常考的知识点之一，这题有点类似于开放题，你回答出越多方法当然越加分，表明你对 Vue 掌握的越熟练。Vue 组件间通信只要指以下 3 类通信：父子组件通信、隔代组件通信、兄弟组件通信，下面我们分别介绍每种通信方式且会说明此种方法可适用于哪类组件间通信。

**（1）props / $emit 适用 父子组件通信**

这种方法是 Vue 组件的基础，相信大部分同学耳闻能详，所以此处就不举例展开介绍。

**（2）ref 与 children 适用 父子组件通信**

ref：如果在普通的 DOM 元素上使用，引用指向的就是 DOM 元素；如果用在子组件上，引用就指向组件children：访问父 / 子实例

**（3）EventBus （on） 适用于 父子、隔代、兄弟组件通信**

这种方法通过一个空的 Vue 实例作为中央事件总线（事件中心），用它来触发事件和监听事件，从而实现任何组件间的通信，包括父子、隔代、兄弟组件。

**（4）listeners 适用于 隔代组件通信（了解）**

attrs" 传入内部组件。通常配合 inheritAttrs 选项一起使用。

listeners" 传入内部组件

**（5）provide / inject 适用于 隔代组件通信**

祖先组件中通过 provider 来提供变量，然后在子孙组件中通过 inject 来注入变量。provide / inject API 主要解决了跨级组件间的通信问题，不过它的使用场景，主要是子组件获取上级组件的状态，跨级组件间建立了一种主动提供与依赖注入的关系。

**（6）Vuex 适用于 父子、隔代、兄弟组件通信**

Vuex 是一个专为 Vue.js 应用程序开发的状态管理模式。每一个 Vuex 应用的核心就是 store（仓库）。“store” 基本上就是一个容器，它包含着你的应用中大部分的状态 ( state )。

Vuex 的状态存储是响应式的。当 Vue 组件从 store 中读取状态的时候，若 store 中的状态发生变化，那么相应的组件也会相应地得到高效更新。

改变 store 中的状态的唯一途径就是显式地提交 (commit) mutation。这样使得我们可以方便地跟踪每一个状态的变化。

## ****6、你使用过 Vuex 吗？（10）****

Vuex 是一个专为 Vue.js 应用程序开发的状态管理模式。每一个 Vuex 应用的核心就是 store（仓库）。“store” 基本上就是一个容器，它包含着你的应用中大部分的状态 ( state )。

（1）Vuex 的状态存储是响应式的。当 Vue 组件从 store 中读取状态的时候，若 store 中的状态发生变化，那么相应的组件也会相应地得到高效更新。

（2）改变 store 中的状态的唯一途径就是显式地提交 (commit) mutation。这样使得我们可以方便地跟踪每一个状态的变化。

主要包括以下几个模块：

State：定义了应用状态的数据结构，可以在这里设置默认的初始状态。

Getter：允许组件从 Store 中获取数据，mapGetters 辅助函数仅仅是将 store 中的 getter 映射到局部计算属性。

Mutation：是唯一更改 store 中状态的方法，且必须是同步函数。

Action：用于提交 mutation，而不是直接变更状态，可以包含任意异步操作。

Module：允许将单一的 Store 拆分为多个 store 且同时保存在单一的状态树中。

## ****7、vue-router 路由模式有几种？（9）****

vue-router 有 3 种路由模式：hash、history、abstract，对应的源码如下所示：

switch (mode) {

其中，3 种路由模式的说明如下：

hash: 使用 URL hash 值来作路由。支持所有浏览器，包括不支持 HTML5 History Api 的浏览器；

history : 依赖 HTML5 History API 和服务器配置。具体可以查看 HTML5 History 模式

abstract : 支持所有 JavaScript 运行环境，如 Node.js 服务器端。如果发现没有浏览器的 API，路由会自动强制进入这个模式.

## ****8、虚拟 DOM 实现原理？（6）****

虚拟 DOM 的实现原理主要包括以下 3 部分：

用 JavaScript 对象模拟真实 DOM 树，对真实 DOM 进行抽象；

diff 算法 — 比较两棵虚拟 DOM 树的差异；

pach 算法 — 将两个虚拟 DOM 对象的差异应用到真正的 DOM 树。

## ****9、Vue 中的 key 有什么作用？（4）****

key 是为 Vue 中 vnode 的唯一标记，通过这个 key，我们的 diff 操作可以更准确、更快速。

【Vue 的 diff 过程可以概括为：oldCh 和 newCh 各有两个头尾的变量 oldStartIndex、oldEndIndex 和 newStartIndex、newEndIndex，它们会新节点和旧节点会进行两两对比，即一共有4种比较方式：newStartIndex 和oldStartIndex 、newEndIndex 和 oldEndIndex 、newStartIndex 和 oldEndIndex 、newEndIndex 和 oldStartIndex，如果以上 4 种比较都没匹配，如果设置了key，就会用 key 再进行比较，在比较的过程中，遍历会往中间靠，一旦 StartIdx > EndIdx 表明 oldCh 和 newCh 至少有一个已经遍历完了，就会结束比较。具体有无 key 的 diff 过程，可以查看作者写的另一篇详解虚拟 DOM 的文章《深入剖析：Vue核心之虚拟DOM》】

所以 Vue 中 key 的作用是：key 是为 Vue 中 vnode 的唯一标记，通过这个 key，我们的 diff 操作可以更准确、更快速

更准确：因为带 key 就不是就地复用了，在 sameNode 函数 a.key === b.key 对比中可以避免就地复用的情况。所以会更加准确。

## ****10、你有对 Vue 项目进行哪些优化？(写出部分即可）（6）****

如果没有对 Vue 项目没有进行过优化总结的同学，可以参考本文作者的另一篇文章《 Vue 项目性能优化 — 实践指南 》，文章主要介绍从 3 个大方面，22 个小方面详细讲解如何进行 Vue 项目的优化。

（1）代码层面的优化

v-if 和 v-show 区分使用场景

computed 和 watch 区分使用场景

v-for 遍历必须为 item 添加 key，且避免同时使用 v-if

长列表性能优化

事件的销毁

图片资源懒加载

路由懒加载

第三方插件的按需引入

优化无限列表性能

服务端渲染 SSR or 预渲染

（2）Webpack 层面的优化

Webpack 对图片进行压缩

减少 ES6 转为 ES5 的冗余代码

提取公共代码

模板预编译

提取组件的 CSS

优化 SourceMap

构建结果输出分析

Vue 项目的编译优化

（3）基础的 Web 技术的优化

开启 gzip 压缩

浏览器缓存

CDN 的使用

使用 Chrome Performance 查找性能瓶颈

### 11.**vue-router有哪几种导航钩子？以及它的参数？（6）**

三种，一是全局导航钩子：router.beforeEach(to,from,next)，作用：跳转前进行判断拦截。

第二种：组件内的钩子

第三种：单独路由独享组件

beforeRouteEnter、afterEnter、beforeRouterUpdate、beforeRouteLeave

参数：有to（去的那个路由）、**from**（离开的路由）、**next**（一定要用这个函数才能去到下一个路由，如果不用就拦截）最常用就这几种

### 12.第一次页面加载会触发哪几个钩子（4）

1. beforeCreate (创建前)
2. created （创建后）
3. beforeMount (挂载前)
4. mounted (挂载后)

13.$route 和 $router 的区别（8）

$route为当前router跳转对象里面可以获取name、path、query、params等

r o u t e r 为 V u e R o u t e r 实 例 ， 想 要 导 航 到 不 同 U R L ， 则 使 用 router为VueRouter实例，想要导航到不同URL，则使用router为VueRouter实例，想要导航到不同URL，则使用router.push方法

返回上一个history也是使用$router.go方法

14.[说说你对router-link的了解](https://blog.csdn.net/weixin_43680581/article/details/104372295" \t "https://www.cnblogs.com/qwer123dsada/p/_blank)（4）

vue-router插件的其中一个组件, 用于跳转路由, 类似于a标签, 它一般也会渲染成a标签, 但是可以通过tag来变更默认渲染元素, 通过to来跳转

15.如何给vue自定义组件添加点击事件？（3）

需要在@click后面加上.native,官方对于native的解释为：

.native -——监听组件根元素的原生事件