前端面试题-VUE专项，共计58道8247字

vue篇

1、什么是MVVM

> MVVM 是 Model-View-ViewModel 的缩写。mvvm 是一种设计思想。Model 层代表数据模型，也可以在 Model 中定义数据修改和操作的业务逻辑；View 代表 UI 组件，它负责将数据模型转化成 UI 展现出来，ViewModel 是一个同步 View 和 Model 的对象。

2、Vue声明周期

> Vue生命周期会经过八个阶段：
>
> 1、beforeCreate(创建前)
>
> 2、created(创建后)
>
> 3、beforeMount(载入前)
>
> 4、mounted(载入后)
>
> 5、beforeUpdate(更新前)
>
> 6、updated（更新后)
>
> 7、beforeDestroy(销毁前)
>
> 8、destroyed（销毁后）

3、为什么vue中data必须是一个函数

> 对象为引用类型，当重用组件时，由于数据对象都指向同一个data对象，当在一个组件中修改data时，其他重用的组件中的data会同时被修改；而使用返回对象的函数，由于每次返回的都是一个新对象（Object的实例），引用地址不同，则不会出现这个问题

4、vue-router有几种导航钩子

> 1、全局导航钩子
>
> 2、组件内的钩子
>
> 3、单独路由独享组件

5、Vue的v-show和v-if区别

> v-if直接影响组件是否被渲染
> v-show是决定元素display的值是不是none
>
> 当需要在显示与隐藏之间进行频繁的切换操作时，就使用v-show。
>
> 当只有一次切换时，我们就使用v-if。

6、vue-loader是什么？使用它的用途有哪些

> vue-loader是解析.vue文件的一个加载器，跟template/js/style转换成js模块，使得.vue文件可以被浏览器解析

7、计算属性和watch的区别

> computed计算属性，依赖其他的属性值，并且computed的属性值有缓存属性，当属性值变化的时候，下一次获取computed属性的时候才会重新计算computed的值。
>
> watch是一种观察的作用，用于监听某些数据的回调。每当所监听的数据发生变化时才能执行回调处理后续操作
>
> 计算属性可以一对多，而watch是一对一

8、prop是什么

> prop是共给父组件给子组件传值得一个重要属性，需要在子组件内规划好该组件需要得props以及每个prop数据格式默认值等等

9、vue 组件通信

> 父传递子：
>
> 父：自定义属性名 + 数据（要传递）=> :value="数据"
>
> 子：props ["父组件上的自定义属性名"] =>进行数据接收)
>
> 子传递父：
>
> 在父组件中注册子组件并在子组件标签上绑定自定义事件的监听。
>
> 子：`this.$emit`('自定义事件名称', 数据) 子组件标签上绑定@自定义事件名称='回调函数'
>
> 父：methods: {自定义事件() {//逻辑处理} }
>
> 兄弟组件：
>
> 通过中央通信 let bus = new Vue()
>
> vuex可以满足任何场景通信需求

10、vue路由传参数有几种方式

> 1.使用query方法传入的参数使用`this.$route.query`接受
>
> 2.使用params方式传入的参数使用`this.$route.params`接受

11、query传参和params传参有什么区别

> 1.params传参可以提前在路由离定义好成为路由的一部分而query不需要
>
> 2.params传参或存在参数刷新丢失的情况而query不会

12、vuex 是什么？ 有哪几种属性

> Vuex 是一个专为 Vue.js 应用程序开发的状态管理模式。
>
> 它有 5 种属性，分别是 state、getter、mutation、action、module

13、vuex 的 store 是什么

> vuex 就是一个仓库，仓库里放了很多对象。其中 state 就是数据源存放地，对应于一般 vue 对象里面的 datastate 里面存放的数据是响应式的，vue 组件从 store 读取数据，若是 store 中的数据发生改变，依赖这相数据的组件也会发生更新它通过 mapState 把全局的 state 和 getters 映射到当前组件的 computed 计算属性

14、vuex 的 mutation 有什么使用技巧

> mutation 里不能进行异步操作，mutation 提交的是对store 数据的更改，一般调用mutation 的都是action ，action 类似于 muation, 不同在于：action 提交的是 mutation,而不是直接变更 store 状态，action 可以包含任意异步操作

15、如何让CSS只在当前组件中起作用

> 将当前组件的`<style>`添加为`scoped`

16、scoped的原理是什么

> 添加`scoped`属性的组件，会给HTML的DOM节点加一个不重复属性标志唯一性，实现类似于“作用域”的作用，不影响全局，这样添加的样式就是给这个唯一标示添加，达到样式隔离的效果

17、keep-alive的作用是什么

> `<keep-alive></keep-alive>`包裹动态组件时，会缓存不活动的组件实例,主要用于保留组件状态或避免重新渲染，实现缓存组件

18、delete和Vue.delete删除数组的区别

> delete只是被删除的元素变成了 empty/undefined 其他的元素的索引还是不变。
> Vue.delete直接删除了数组 改变了数组的长度。

19、`$nextTick`是什么

> `$nextTick` 是在下次 DOM 更新循环结束之后执行延迟回调，可以保证回调函数一定实在DOM更新后执行的

20、v-on 常用修饰符

> .stop 阻止事件向上冒泡。
>
> .prevent 阻止当前事件的默认行为
>
> .self 事件绑定的元素本身触发时才触发回调
>
> .once 绑定的事件只会被触发一次

21、v-for 为什么需要绑定Key

> 当Vue用 v-for 正在更新已渲染过的元素列表是，它默认用“就地复用”策略。如果数据项的顺序被改变，Vue将不是移动DOM元素来匹配数据项的改变，而是简单复用此处每个元素，并且确保它在特定索引下显示已被渲染过的每个元素。
> 为每项提供一个唯一 key 属性，在 Vue 的虚拟 DOM 算法里，在新旧 nodes 对比时辨识 VNodes。如果不使用 key，Vue 会使用一种最大限度减少动态元素并且尽可能的尝试修复/再利用相同类型元素的算法。使用 key，它会基于 key 的变化重新排列元素顺序，并且会移除 key 不存在的元素。

22、v-for 与 v-if 的优先级

> v-for比v-if优先，如果每一次都需要遍历整个数组，将会影响速度

23、vue中的 ref 是什么

> ref 被用来给元素或子组件注册引用信息。引用信息将会注册在父组件的 $refs 对象上。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，引用就指向组件实例

24、Vue的路由hash模式 和 history模式的区别

> hash模式在浏览器中有个符号“#”，#以及#后面的字符称之为hash，用window.location.hash读取而history是采用HTML5的新特性，底层使用pushState（），replaceState（）可以对浏览器历史记录栈进行修改，以及popState事件的监听到状态变更，history 模:前端的 URL 必须和实际向后端发起请求的 URL 一致，后端如果缺少对 /items/id 的路由处理，将返回 404 错误。

25、 $route和$router的区别

> $route是“路由信息对象”，包括path，params，hash，query，fullPath，matched，name等路由信息参数。
>
> $router是'路由实例'对象包括了路由的跳转方法，钩子函数等

26、vue如何兼容ie的问题

> vue本身不兼容IE10一下的，但是可以使用babel-polyfill插件改善兼容情况

27、如何优化SPA应用的首屏加载速度慢的问题

> 1.将公用的JS库通过script标签外部引入，减小 app.bundel 的大小，让浏览器并行下载资源文件，提高下载速度；
>
> 2.在配置 路由时，页面和组件使用懒加载的方式引入，进一步缩小 app.bundel 的体积，在调用某个组件时再加载对应的js文件；
>
> 3.加一个首屏loading图，提升用户体验；
>
> 4.使用预渲染插件prerender-spa-plugin生成对特定路由静态的html文件

28、Vue 改变数组有时候无法触发视图更新是什么原因

> Vue是通过Object.defineProperty()来实现双向数据绑定的。把一个普通 JavaScript 对象传给 Vue 实例的 data 选项，Vue 将遍历此对象所有的属性，并使用 Object.defineProperty 把这些属性全部转为 getter/setter，当使用push(), pop(), shift(), unshift(), splice(), sort(), reverse()等数组原生方法操作数据的手可以引发页面更新，但是如果直接通过索引更改数组内容就会有问题，这个时候可以通过Vue.set解决

29、Vue中双向数据绑定是如何实现的

> vue 双向数据绑定是通过 数据劫持 结合 发布订阅模式的方式来实现的， 也就是说数据和视图同步，数据发生变化，视图跟着变化，视图变化，数据也随之发生改变；vue双向数据绑定，其核心是 Object.defineProperty()方法

30、单页面应用和多页面应用区别及优缺点

> 单页面应用就是指只有一个主页面的应用，浏览器一开始要加载所有必须的 html, js, css。所有的页面内容都包含在这个所谓的主页面中。但在写的时候，还是会分开写（页面片段），然后在交互的时候由路由程序动态载入，单页面的页面跳转，仅刷新局部资源。多应用于pc端。
> 多页面是指一个应用中有多个页面，页面跳转时是整页刷新.
>
> 单页面的优点是用户体验好，快，内容的改变不需要重新加载整个页面，基于这一点spa对服务器压力较小；前后端分离；页面效果会比较炫酷（比如切换页面内容时的专场动画）。
> 单页面缺点是不利于seo；导航不可用，如果一定要导航需要自行实现前进、后退。（由于是单页面不能用浏览器的前进后退功能，所以需要自己建立堆栈管理）；初次加载时耗时多；页面复杂度提高很多

31、文件夹assets和static的区别

> assets和static两个都是存放静态资源文件，但是assets中存放的静态资源文件在项目打包时会进行编译，而static不会

32、RouterLink在IE和Firefox中不起作用（路由不跳转）的问题

> 方法一：只用a标签，不适用button标签
>
> 方法二：使用button标签和Router.navigate方法

33、vue获取数据在哪个周期函数

> 在 created/beforeMount/mounted 中都可以
>
> 如果不需要等待页面渲染完毕最好就在created里请求

34、简述vue中diff算法原理

diff算法是一种优化手段，将前后两个模块进行差异化对比，修补（更新）差异的过程叫做patch(打补丁)，从以下几点来理解：

1. 当数据发生变化时，vue是怎么更新节点的？

要知道渲染真实DOM的开销是很大的，比如有时候我们修改了某个数据，如果直接渲染到真实dom上会引起整个dom树的重绘和重排，有没有可能我们只更新我们修改的那一小块dom而不要更新整个dom呢？diff算法能够帮助我们。

我们先根据真实DOM生成一颗 virtual DOM ， 当 virtual DOM 某个节点的数据改变后会生成一个新的 Vnode ，然后 Vnode 和 oldVnode 作对比，发现有不一样的地方就直接修改在真实的DOM上，然后使 oldVnode 的值为 Vnode 。

diff的过程就是调用名为 patch 的函数，比较新旧节点，一边比较一边给 真实的DOM 打补丁。

2. virtual DOM和真实DOM的区别？

virtual DOM是将真实的 DOM 的数据抽取出来，以对象的形式模拟树形结构，diff算法比较的也是 virtual DOM

3. diff的比较方式？

在采取diff算法比较新旧节点的时候，比较只会在同层级进行, 不会跨层级比较。

35、Vue3.0 里为什么要用 Proxy API 替代 defineProperty API?

响应式优化。

a. **defineProperty API 的局限性最大原因是它只能针对单例属性做监听。**

Vue2.x 中的响应式实现正是基于 defineProperty 中的 descriptor，对 data 中的属性做了遍

历 + 递归，为每个属性设置了 getter、setter。

这也就是为什么 Vue 只能对 data 中预定义过的属性做出响应的原因，在 Vue 中使用下标的方式直接修改属性的值或者添加一个预先不存在的对象属性是无法做到 setter 监听的，这是 defineProperty 的局限性。

b. **Proxy API 的监听是针对一个对象的，那么对这个对象的所有操作会进入监听操作，这就完全可以代理所有属性，将会带来很大的性能提升和更优的代码。**

Proxy 可以理解成，在目标对象之前架设一层"拦截"，外界对该对象的访问，都必须先通过这层拦截，因此提供了一种机制，可以对外界的访问进行过滤和改写。

c. 响应式是惰性的

在 Vue.js 2.x 中，对于一个深层属性嵌套的对象，要劫持它内部深层次的变化，就需要递归遍历这个对象，**执行 Object.defineProperty 把每一层对象数据都变成响应式的，这无疑会有很大的性能消耗。**

在 Vue.js 3.0 中，使用 Proxy API 并不能监听到对象内部深层次的属性变化，因此它的处理方式是在 getter 中去递归响应式，这样的好处是**真正访问到的内部属性才会变成响应式，简单的可以说是按需实现响应式，减少性能消耗。**

36、Vue3.0 编译做了哪些优化?

a. 生成 Block tree

Vue.js 2.x 的数据更新并触发重新渲染的粒度是组件级的，单个组件内部 需要遍历该组件的整个 vnode 树。**在 2.0 里，渲染效率的快慢与组件大小成正相关：组件越大，渲染效率越慢。并且，对于一些静态节点，又无数据更新，这些遍历都是性能浪费。**

Vue.js 3.0 做到了通过编译阶段对静态模板的分析，编译生成了 Block tree。 **Block tree 是一个将模版基于动态节点指令切割的嵌套区块**，每个 区块内部的节点结构是固定的，每个区块只需要追踪自身包含的动态节点。所以，**在 3.0 里，渲染效率不再与模板大小成正相关，而是与模板中动态节点的数量成正相关**

b. slot 编译优化

Vue.js 2.x 中，如果有一个组件传入了 slot，那么每次父组件更新的时候，会强制使子组件 update，造成性能的浪费。

Vue.js 3.0 优化了 slot 的生成，使得非动态 slot 中属性的更新只会触发子组件的更新。**动态 slot 指的是在 slot 上面使用 v-if，v-for，动态 slot 名字等会导致 slot 产生运行时动态变化但是又无法被子组件 track 的操作。**

c. diff 算法优化

37、Vue3.0 新特性 —— Composition API 与 React.js 中 Hooks 的异同点

a. React.js 中的 Hooks 基本使用

React Hooks□允许你 "勾入"□诸如组件状态和副作用处理等 React□功能中。Hooks□只能用在函数组件中，并允许我们在不需要创建类的情况下将状态、副作用处理和更多东西带入组件中。

React 核心团队奉上的采纳策略是不反对类组件，所以你可以升级 React□版本、在新组件中开始尝试 Hooks，并保持既有组件不做任何更改。

useState 和 useEffect 是 React Hooks 中的一些例子，使得函数组件中也能增加状态和运行副作用。

我们也可以自定义一个 Hooks，它打开了代码复用性和扩展性的新大门。

b. Vue Composition API 基本使用

Vue Composition API 围绕一个新的组件选项 setup 而创建。setup() 为 Vue 组件提供了状态、计算值、watcher 和生命周期钩子。

并没有让原来的 API（Options-based API）消失。允许开发者 结合使用新旧两种 API（向下兼容）。

c. 原理

React hook 底层是基于链表实现，调用的条件是每次组件被 render 的时候都会顺序执行所有的 hooks。

Vue hook 只会被注册调用一次，Vue 能避开这些麻烦的问题，原因在于它对数据的响应是基于 proxy 的，对数据直接代理观察。（这种场景下，只要任何一个更改 data 的地方，相关的 function 或者 template 都会被重新计算，因此避开了 React 可能遇到的性能上的问题）。

React 中，数据更改的时候，会导致重新 render，重新 render 又会重新把 hooks 重新注册一次，所以 React 复杂程度会高一些。

38、Vue3.0 是如何变得更快的?

a. diff 方法优化

Vue2.x 中的虚拟 dom 是进行全量的对比。

Vue3.0□中新增了静态标记（PatchFlag）：在与上次虚拟结点进行对比的时候，值对比带有 patch flag 的节点，并且可以通过 flag 的信息得知当前节点要对比的具体内容化。**b. hoistStatic 静态提升**

Vue2.x : 无论元素是否参与更新，每次都会重新创建。

Vue3.0 : 对不参与更新的元素，只会被创建一次，之后会在每次渲染时候被不停的复用。

c. cacheHandlers 事件侦听器缓存

默认情况下 onClick 会被视为动态绑定，所以每次都会去追踪它的变化但是因为是同一个函数，所以没有追踪变化，直接缓存起来复用即可。

39、为什么虚拟 dom 会提高性能?

虚拟 dom 相当于在 js 和真实 dom 中间加了一个缓存，利用 dom diff 算法避免了没有必要的 dom 操作，从而提高性能。

具体实现步骤如下：

1.用 JavaScript 对象结构表示 DOM 树的结构；然后用这个树构建一个真正的 DOM 树，插到文档当中；

2.当状态变更的时候，重新构造一棵新的对象树。然后用新的树和旧的树进行比较，记录两棵树差异；

把 2 所记录的差异应用到步骤 1 所构建的真正的 DOM 树上，视图就更新了。

40、Vue的父子组件生命周期钩子函数执行顺序

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

41、v-model 原理

v-model 只是语法糖而已。
v-model 在内部为不同的输入元素使用不同的 property 并抛出不同的事件。
text 和 textarea 元素使用 value property 和 input 事件；
checkbox 和 radio 使用 checked property 和 change事件；
select 字段将 value 作为 prop 并将 change 作为事件。
注意：对于需要使用输入法的语言，你会发现 v-model 不会在输入法组合文字过程中得到更新。
在普通元素上：
input v-model='sth'
input v-bind:value='sth' v-on:input='sth = $event.target.value'

42、Vue事件绑定原理

原生事件绑定是通过 addEventListener 绑定给真实元素的，组件事件绑定是通过Vue自定义的$on实现的。如果要在组件上使用原生事件，需要加.native修饰符，这样就相当于在父组件中把子组件当做普通的HTML标签，然后加上原生事件。
o n 、 on、on、emit 是基于发布订阅模式的，维护一个事件中心，on的时候将事件按名称存在事件中心里，称之为订阅者，然后emit将对应的事件进行发布，去执行事件中心里的对应的监听器。

43、vue-router 路由钩子函数是什么？执行顺序是什么？

路由钩子的执行流程，钩子函数种类有：全局守卫、路由守卫、组件守卫。
完整的导航解析流程：
1、导航被触发。
2、在失活的组件里调用 beforeRouterLeave 守卫。
3、调用全局的 beforeEach 守卫。
4、在重用的组件调用 beforeRouterUpdate 守卫（2.2+）。
5、在路由配置里面 beforeEnter。
6、解析异步路由组件。
7、在被激活的组件里调用 beforeRouterEnter。
8、调用全局的 beforeResolve 守卫（2.5+）。
9、导航被确认。
10、调用全局的 afterEach 钩子。
11、触发 DOM 更新。
12、调用 beforeRouterEnter 守卫中传给next的回调函数，创建好的组件实例会作为回调函数的参数传入。

44、Vuex 页面刷新数据丢失怎么解决?

需要做 vuex 数据持久化，一般使用本地储存的方案来保存数据，可以自己设计存储方案，也可以使用第三方插件。
推荐使用 vuex-persist (脯肉赛斯特)插件，它是为 Vuex 持久化储存而生的一个插件。不需要你手动存取 storage，而是直接将状态保存至 cookie 或者 localStorage中。

或使用pinia解决

45、Vuex 为什么要分模块并且加命名空间?

模块： 由于使用单一状态树，应用的所有状态会集中到一个比较大的对象。当应用变得非常复杂时，store 对象就有可能会变得相当臃肿。为了解决以上问题，Vuex 允许我们将 store 分割成模块（module）。每个模块拥有自己的 state、mutation、action、getter、甚至是嵌套子模块。

命名空间： 默认情况下，模块内部的 action、mutation、getter是注册在全局命名空间的 — 这样使得多个模块能够对同一 mutation 或 action 做出响应。如果希望你的模块具有更高的封装度和复用性，你可以通过添加 namespaced:true 的方式使其成为带命名的模块。当模块被注册后，他所有 getter、action、及 mutation 都会自动根据模块注册的路径调整命名。

46、使用过 Vue SSR 吗？说说 SSR

SSR 也就是服务端渲染，也就是将 Vue 在客户端把标签渲染成 HTML 的工作放在服务端完成，然后再把 html 直接返回给客户端。

优点：
SSR 有着更好的 SEO、并且首屏加载速度更快。
缺点：
开发条件会受限制，服务器端渲染只支持 beforeCreate 和 created 两个钩子，当我们需要一些外部扩展库时需要特殊处理，服务端渲染应用程序也需要处于 Node.js 的运行环境。
服务器会有更大的负载需求。

47、vue 中使用了哪些设计模式?

1、工厂模式 - 传入参数即可创建实例
虚拟 DOM 根据参数的不同返回基础标签的 Vnode 和组件 Vnode。
2、单例模式 - 整个程序有且仅有一个实例
vuex 和 vue-router 的插件注册方法 install 判断如果系统存在实例就直接返回掉。
3、发布-订阅模式。（vue 事件机制）
4、观察者模式。（响应式数据原理）
5、装饰器模式（@装饰器的用法）
6、策略模式，策略模式指对象有某个行为，但是在不同的场景中，该行为有不同的实现方案 - 比如选项的合并策略。

48、你都做过哪些 Vue 的性能优化?

这里只列举针对 Vue 的性能优化，整个项目的性能优化是一个大工程。

对象层级不要过深，否则性能就会差。
不需要响应式的数据不要放在 data 中
v-if 和 v-show 区分使用场景
computed 和 watch 区分场景使用
v-for 遍历必须加 key，key最好是id值，且避免同时使用 v-if
大数据列表和表格性能优化 - 虚拟列表 / 虚拟表格
防止内部泄露，组件销毁后把全局变量和时间销毁
图片懒加载
路由懒加载
异步路由
第三方插件的按需加载
适当采用 keep-alive 缓存组件
防抖、节流的运用
服务端渲染 SSR or 预渲染

49、Vue.mixin 的使用场景和原理

在日常开发中，我们经常会遇到在不同组件中经常用到一些相同或者相似的代码，这些代码的功能相对独立，可以通过vue 的 mixin 功能抽离公共的业务逻辑，原理类似“对象的继承”，当组件初始化时会调用 mergeOptions 方法进行合并，采用策略模式针对不同的属性进行合并。当组件和混入对象含有相同名选项时，这些选项将以恰当的方式进行“合并”。

50、nextTick 使用场景和原理

nextTick 中的回调是在下次 DOM 更新循环结束之后执行的延迟回调。在修改数据之后立即使用这个方法，获取更新后的 DOM。主要思路就是采用微任务优先的方式调用异步方法去执行 nextTick 包装的方法。

51、keep-alive 使用场景和原理

keep-alive 是 Vue 内置的一个组件，可以实现组件缓存，当组件切换时不会对当前组件进行卸载。

常用的两个属性 include/exclude，允许组件有条件的进行缓存。
两个生命周期 activated/deactivated，用来得知当前组件是否处理活跃状态。
keep-alive 运用了 LRU 算法，选择最近最久未使用的组件予以淘汰。

52、Vue.set 方法原理

了解 Vue 响应式原理的同学都知道在两种情况下修改 Vue 是不会触发视图更新的。
1、在实例创建之后添加新的属性到实例上（给响应式对象新增属性）
2、直接更改数组下标来修改数组的值。
Vue.set 或者说是 $set 原理如下
因为响应式数据 我们给对象和数组本身新增了**ob**属性，代表的是 Observer 实例。当给对象新增不存在的属性，首先会把新的属性进行响应式跟踪 然后会触发对象 ob 的dep收集到的 watcher 去更新，当修改数组索引时我们调用数组本身的 splice 方法去更新数组。

53、Vue.extend 作用和原理

官方解释：Vue.extend 使用基础 Vue 构造器，创建一个“子类”。参数是一个包含组件选项的对象。

其实就是一个子类构造器，是Vue组件的核心api。实现思路就是使用原型继承的方法返回了 vue 的子类，并且利用 mergeOptions 把传入组件的 options 就和父类的 options 进行了合并。

54、写过自定义指令吗？原理是什么？

指令本质上是装饰器，是 vue 对 HTML 元素的扩展，给 HTML 元素添加自定义功能。vue 编译 DOM 时，会找到指令对象，执行指令的相关方法。

自定义指令有五个生命周期（也叫钩子函数），分别是 bind、inserted、update、componentUpdated、unbind

1、bind：只调用一次，指令第一次绑定到元素时调用。在这里可以进行一次性的初始化设置。
2、inserted：被绑定元素插入父节点时调用。
3、update：被绑定元素所在的模板更新时调用，而不论绑定值是否变化。通过比较前后的绑定值。
4、componentUpdated：被绑定元素所在模板完成一次更新周期时调用。
5、unbind：只调用一次，指令与元素解绑时调用。
原理：
1、在生成 ast 语法树时，遇到指令会给当前元素添加 directives 属性
2、通过 genDirectives 生成指令代码
3、在 patch 前将指令的钩子提取到 cbs 中，在 patch 过程中调用对应的钩子。
4、当执行指令对应钩子函数时，调用对应指令定义方法。

55、Vue 修饰符有哪些?

事件修饰符
.stop 阻止事件继续传播
.prevent 阻止标签默认行为
.capture 使用事件捕获模式，即元素自身触发的事件先在此处处理，然后才交由内部元素进行处理
.self 只当在 event.target 是当前元素自身时触发处理函数
.once 事件只会触发一次
.passive 告诉浏览器你不想阻止事件的默认行为
v-model 的修饰符
.lazy 通过这个修饰符，转变为在 change 事件再同步
.number 自动将用户输入值转化为数值类型
.trim 自动过滤用户输入的收尾空格

键盘事件修饰符
.enter
.tab
.delete (捕获“删除”和“退格”键)
.esc
.space
.up
.down
.left
.right
系统修饰符
.ctrl
.alt
.shift
.meta
鼠标按钮修饰符
.left
.right
.middle

56、Vue 模板编译原理

Vue 的编译过程就是将 template 转化为 render 函数的过程，分为以下三步：
第一步是将 模板字符串转换成 element ASTs（解析器）
第二步是对 AST 进行静态节点标记，主要用来做虚拟 DOM 的渲染优化（优化器）
第三步是 使用element ASTs 生成 render 函数代码字符串（代码生成器）

57、生命周期钩子是如何实现的

Vue 的生命周期钩子核心实现是利用发布订阅模式先把用户传入的生命周期钩子订阅好（内部采用数组的方法存储）然后在创建组件实例的过程中会一次执行对应的钩子方法（发布）

58、能说下 vue-router 中常用的路由模式和实现原理吗?

hash 模式
1、location.has 的值实际就是 URL 中 # 后面的东西。它的特点在于：hash虽然出现 URL 中，但不会被包含在 HTTP 请求中，对后端完全没有影响，因此改变 hash 不会重新加载页面。

2、可以为 hash 的改变添加监听事件

window.addEventListener("hashchange",funcRef,false)

每一次改变 hash (window.location.hash)，都会在浏览器的访问历史中增加一个记录，利用hash的以上特点，就可以实现前端路由“更新视图但不重新请求页面”的功能了
特点：兼容性好但是不美观

history 模式
利用 HTML5 History Interface 中新增的 pushState() 和 replaceState() 方法。

这两个方法应用于浏览器的历史记录站，在当前已有的 back、forward、go 的基础上，他们提供了对历史记录进行修改的功能。这两个方法有个共同点：当调用他们修改浏览器历史记录栈后，虽然当前 URL 改变了，但浏览器不会刷新页面，这就为单页面应用前端路由“更新视图但不重新请求页面”提供了基础

特点：虽然美观，但是刷新会出现 404 需要后端进行配置。