Vue的双向数据绑定原理是什么？或者vue的响应式原理

vue.js是采用数据劫持（Observer），并结合发布者——订阅者的模式：通过Object.defineProperty()来劫持vue中各个属性的setter、getter，在数据变动的时候，再发布消息给订阅者。

要实现mvvm的双向绑定，就必须要实现以下几点：

1、实现一个数据监听器Observer（发布者），能够对数据对象的所有属性进行监听，如有变动可拿到最新值并通知订阅者

2、实现一个指令解析器Compile，对每个元素节点的指令进行扫描和解析，根据指令模板替换数据，以及绑定相应的更新函数

3、实现一个Watcher（订阅者），作为连接Observer和Compile的桥梁，能够订阅并收到每个属性变动的通知，执行指令绑定的相应回调函数，从而更新视图

mvvm框架是什么？它和其它框架（jquery）的区别是什么？哪些场景适合？

一个model+view+viewModel框架，数据模型model，viewModel连接两个

区别：vue数据驱动，通过数据来显示视图层而不是节点操作。

场景：数据操作比较多的场景，更加便捷

**v-model是什么？怎么使用？**

使用v-model指令，实现视图和数据的双向绑定。

v-model只能用于表单元素上：input\textarea\select标签，

组件也可以定制v-model实现双向绑定。

**vuex是什么？怎么使用？哪种功能场景使用它？**

vue框架中状态管理插件。适合在大型前端应用中存在多处组件公用数据源的场景。

**自定义指令的方法有哪些钩子函数？还有哪些钩子函数参数**

**Vue.directive(name, data)和directives:{}**

**5个钩子函数，传入4个参数**

**Bind,inserted,update,componentUpdated,unbind**

**El,data,vnode,oldVnode**

**vue-router是什么？它有哪些组件？**

vue用来写路由一个插件。router-link、router-view

**请详细说下你对vue生命周期的理解？**

**总共分为8个阶段创建前/后，载入前/后，更新前/后，销毁前/后。**

**创建前/后： 在beforeCreated阶段，vue实例的挂载元素$el和数据对象data都为undefined，还未初始化。在created阶段，vue实例的数据对象data有了，$el还没有。**

**载入前/后：在beforeMount阶段，vue实例的$el和data都初始化了，但还是挂载之前为虚拟的dom节点，data.message还未替换。在mounted阶段，vue实例挂载完成，data.message成功渲染。**

**更新前/后：当data变化时，会触发beforeUpdate和updated方法。**

**销毁前/后：在执行destroy方法后，对data的改变不会再触发周期函数，说明此时vue实例已经解除了事件监听以及和dom的绑定，但是dom结构依然存在**

**我们把 React.js 将组件渲染，并且构造 DOM 元素然后塞入页面的过程称为组件的挂载。**

[Vue原理解析之Virtual Dom](https://segmentfault.com/a/1190000008291645)

DOM是文档对象模型(Document Object Model)的简写，在浏览器中我们可以通过js来操作DOM，但是这样的操作性能很差，于是Virtual Dom应运而生。我的理解，Virtual Dom就是在js中模拟DOM对象树来优化DOM操作的一种技术或思路。

 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

 当状态变更的时候，重新构造一棵新的vnode树。然后用新vnode树和旧vnode树进行比较，记录两棵树差异

 把所记录的差异应用来构建的真正的DOM树，视图就更新了

路由的mode：

hash（默认）：

hash模式背后的原理是onhashchange事件，使用 URL的 hash，模拟一个完整的URL，于是当URL改变时，页面不会重新加载

history

pushState,replaceState两个方法，用onpopstate侦听，丢掉#，但是需要后端配合，不然容易404

导航方法：

Route-link连接，编程式router.push、 replace 、 go、back、forward()

**导航守卫（钩子函数）×7：**

1.全局守卫：

router.beforeEach(to, from, next)：导航触发时，全局前置守卫按照创建顺序调用

router.beforeResolve(to, from, next)：与beforeEach类似，区别是在导航触发时，同时在所有守卫和异步路由组件被解析之后，解析守卫就被调用

router.afterEach(to, from): 导航被确认之后调用

2.路由独享的守卫：你可以在路由配置上直接定义 beforeEnter 守卫：

beforeEnter (to, from, next)：导航被触发后，被确认之前调用

3.组件内的守卫：你可以在路由组件内直接定义以下路由导航守卫：

beforeRouteEnter (to, from, next)：导航被触发后，被确认之前调用，不！能！获取组件实例 this，因为当守卫执行前，组件实例还没被创建

beforeRouteUpdate (to, from, next) ：在当前路由改变，但是该组件被复用时调用

举例来说，对于一个带有动态参数的路径 /foo/:id，在 /foo/1 和 /foo/2 之间跳转的时候

beforeRouteLeave (to, from, next) ：导航离开该组件的对应路由时调用