# Vue面试题

**1.vue中的MVVM模式**

即Model-View-ViewModel。

Vue是以数据为驱动的，Vue自身将DOM和数据进行绑定，一旦创建绑定，DOM和数据将保持同步，每当数据发生变化，DOM会跟着变化。

ViewModel是Vue的核心，它是Vue的一个实例。Vue实例时作用域某个HTML元素上的，这个HTML元素可以是body，也可以是某个id所指代的元素。DOMListeners和DataBindings是实现双向绑定的关键。DOMListeners监听页面所有View层DOM元素的变化，当发生变化，Model层的数据随之变化；DataBindings监听Model层的数据，当数据发生变化，View层的DOM元素随之变化。

**2.v-show指令，v-if的区别**

条件渲染指令，与v-if不同的是，无论v-show的值为true或false，元素都会存在于HTML代码中；而只有当v-if的值为true，元素才会存在于HTML代码中。v-show指令只是设置了元素CSS的style值

**3.如何让css只在当前组件中起作用**

在每一个vue组件中都可以定义各自的css，js，如果希望组件内写的css只对当前组件起作用，只需要在style中写入scoped，即：&lt;stylescoped&gt;&lt;/style&gt;

**4.指令keep-alive**

在vue-router写着keep-alive，keep-alive的含义：如果把切换出去的组件保留在内存中，可以保留它的状态或避免重新渲染。为此可以添加一个keep-alive指令

&lt;component:is='curremtView'keep-alive&gt;&lt;/component&gt;

**5.Vuejs组件**

vuejs构建组件使用

Vue.component('componentName',{/\*component\*/})；这里注意一点，组件要先注册再使用

Vue.component('mine',{

template:'#mineTpl',

props:['name','title','city','content']});

varv=newVue({

el:'#vueInstance',data:{

name:'zhang',

title:'thisistitle',city:'Beijing',

content:'thesearesomedescaboutBlog'}});

**6.路由嵌套**

路由嵌套会将其他组件渲染到该组件内，而不是进行整个页面跳转router-view本身就是将组件渲染到该位置，想要进行页面跳转，就要将页面渲染到根组件，在起始配置路由时

候写到：

varApp=Vue.extend({root});router.start(App,'#app');这里首先将根组件注册进来，用于将路由中配置好的各个页面渲染出来，然后将根组件挂载到与#app匹配的元素上。

**7.指令v-el的使用**

有时候我们想就像使用jquery那样去访问一个元素，此时就可以使用v-el指令，去给这个元素注册一个索引，方便通过所属实例的$el访问这个元素。注意

HTML不区分大小写，所以v-el:someEl将转换为全小写。可以用v-el:some-el然后设置this.$el.someEl。示例

&lt;spanv-el:msg&gt;hello&lt;/span&gt;

&lt;spanv-el:other-msg&gt;world&lt;/span&gt;

this.$els.msg.textContent//-&gt;"hello"

this.$els.otherMsg.textContent//-&gt;"world"this.$els.msg//-&gt;&lt;span&gt;hello&lt;/span&gt;**8.vuejs中使用事件名**

在vuejs中，我们经常要绑定一些事件，有时候给DOM元素绑定，有时候给组件绑定。绑定事件在HTML中用v-on:click-"event",这时evet的名字不要出现大写，因为在1.x中不区分大小写，所以如果我们在HTML写v-on:click="myEvent"而在js中写myEvent就出错误，所以在vuejs的1.x绑定事件时候，要尽量避免使用大写字母。在2.0中没有该限制！

**9.Vue.js是什么**

Vue.js（是一套构建用户界面的渐进式框架。与其他重量级框架不同的是，Vue采用自底向上增量开发的设计。Vue的核心库只关注视图层，并且非常容易学习，非常容易与其它库或已有项目整合。另一方面，Vue完全有能力驱动采用单文件组件和Vue生态系统支持的库开发的复杂单页应用。

Vue.js的目标是通过尽可能简单的API实现响应的数据绑定和组合的视图组件

**10.VueJS特性：**

I:MVVM模式（数据变量（model）发生改变视图（view）也改变，视图（view）改变，数据变量（model）也发生改变）

使用MVVM模式有几大好处：

1.低耦合。View可以独立于Model变化和修改，一个ViewModel可以绑定到不同的View上，当View变化的时候Model可以不变，当Model变化的时候View也可以不变。

2.可重用性。可以把一些视图的逻辑放在ViewModel里面，让很多View重用这段视图逻辑。

3.独立开发。开发人员可以专注与业务逻辑和数据的开发(ViewModel)。设计人员可以专注于界面(View)的设计。

4.可测试性。可以针对ViewModel来对界面(View)进行测试II:组件化III指令系统

IIII:vue2.0开始支持虚拟dom

vue1.0是操作的是真的dom元素而不是虚拟的虚拟dom:可以提升页面的刷新速度

虚拟DOM有利也有弊。

A：大小-其中之一就是更多的功能意味着代码包中更多行的代码。幸运的是，Vue.js2.0依旧比较小（当前版本21.4kb），并

且也正在删除很多东西。

B:内存-同样，虚拟DOM需要将现有的DOM拷贝后保存在内存中，这是一个在DOM更新速度和内存使用中的权衡。

C:并不适用所有情况-如果虚拟DOM可以一次性进行批量的修改是非常好的。但是如果是单独的、稀少的更新呢？这样的任何

DOM更新都将会使虚拟DOM带来无意义的预计算

**11.Vue.js特点**

简洁：页面由HTML模板+Json数据+Vue实例组成数据驱动：自动计算属性和追踪依赖的模板表达式组件化：用可复用、解耦的组件来构造页面轻量：代码量小，不依赖其他库快速：精确有效批量DOM更新

模板友好：可通过npm，bower等多种方式安装，很容易融入

**12.Vue.js和AngularJS之间的区别是什么?**

下面是一些选择Vue而不是Angular的可能原因；

Vue.js是一个更加灵活开放的解决方案。它允许你以希望的方式组织你的应用程序，而不是任何时候都必须遵循Angular制定的规则。它仅仅是一个视图层，所以你可以将它嵌入一个现有页面而不一定要做成一个庞大的单页应用。在结合其他库方面它给了你更大的的空间，但相应，你也需要做更多的架构决策。例如，Vue.js核心默认不包含路由和ajax功能，并且通常假定你在用应用中使用了一个外部的模块构建系统。这可能是最重要的区别在API和内部设计方面，Vue.js比Angular简单得多,因此你可以快速地掌握它的全部特性并投入开发。

Vue.js拥有更好的性能，因为它不使用脏检查。当watcher越来越多时,Angular会变得越来越慢，因为作用域内的每一次数据变更，所有的watcher都需要被重新求值。Vue则根本没有这个问题，因为它采用的是基于依赖追踪的观察系统，所以所有的数据变更触发都是独立的，除非它们之间有明确的依赖关系。Vue.js中指令和组件的概念区分得更为清晰。指令只负责封装DOM操作，而组件代表一个自给自足的独立单元——它拥有自己的视图和数据逻辑。在Angular中它们两者间有不少概念上的混淆。

**13.Vue.js和React.js有什么区别?**

React.js和Vue.js确实有一些相似——它们都提供数据驱动、可组合搭建的视图组件。然而，它们的内部实现是完全不同的。React是基于VirtualDOM——一种在内存中描述DOM树状态的数据结构。React中的数据通常被看作是不可变的，而DOM操作则是通过VirtualDOM的diff来计算的。与之相比，Vue.js中的数据默认是可变的，而数据的变更会直接出发对应的DOM更新。相比于VirtualDOM，Vue.js使用实际的DOM作为模板，并且保持对真实节点的引用来进行数据绑定。

VirtualDOM提供了一个函数式的描述视图的方法，这很cool。因为它不使用数据观察机制，每次更新都会重新渲染整个应用，因此从定义上保证了视图通与数据的同步。它也开辟了JavaScript同构应用的可能性。

实话实说，我自己对React的设计理念也是十分欣赏的。但React有一个问题就是组件的逻辑和视图结合得非常紧密。对于部分开发者来说，他们可能觉得这是个优点，但对那些像我一样兼顾设计和开发的人来说，还是更偏好模板，因为模板能让我们更好地在视觉上思考设计和CSS。JSX和JavaScript逻辑的混合干扰了我将代码映射到设计的思维过程。相反，Vue.js通过在模板中加入一个轻量级的DSL(指令系统)，换来一个依旧直观的模板，且能够将逻辑封装进指令和过滤器中。

React的另一个问题是：由于DOM更新完全交由VirtualDOM管理，当你真的想要自己控制DOM是就有点棘手了（虽然理论上你可以，但这样做时你本质上在对抗React的设计思想）。对于需要复杂时间控制的动画来说这就变成了一项很讨人厌的限制。在这方面，Vue.js允许更多的灵活性，并且有不少用Vue.js构建的富交互实例