

# vue简易题

---

## vue 面试题汇总

---

1、**active-class** 是哪个组件的属性？嵌套路由怎么定义

(1)、**active-class** 是 **vue-router** 模块的 **router-link** 组件的属性  
定义嵌套路由

(2)、使用 **children**

2、怎么定义 **vue-router** 的动态路由？怎么获取传过来的值

在 **router** 目录下的 **index.js** 文件中，对 **path** 属性加上 **/:id**。

使用 **router** 对象的 **params.id** 获取

3、**vue-router** 有哪几种导航钩子？

三种，

(1)、全局导航钩子

**router.beforeEach(to, from, next),**

**router.beforeResolve(to, from, next),**

**router.afterEach(to, from ,next)**

(2)、组件内钩子 **beforeRouteEnter, beforeRouteUpdate, beforeRouteLeave**

(3)、单独路由独享组件

**beforeEnter**

#### 4、**v-model** 是什么？怎么使用？**vue**中标签怎么绑定事件

**v-model** 可以实现双向绑定，

绑定事件：`<input @click="doLog" />`

#### 5、**axios** 是什么？怎么使用？描述使用它实现登录功能的流程

**axios** 是请求后台资源的模块。 `npm i axios -S`

如果发送的是跨域请求，需在配置文件中 `config/index.js` 进行配置

#### 6、**vuex** 是什么？怎么使用？哪种功能场景使用它

**vuex** 是专门为 **vue** 开发的数据状态管理模式。组件之间数据状态共享

使用场景：音乐播放、登录状态、购物车

```
// 新建 store.js
import vue from 'vue'
import vuex from 'vuex'
vue.use(vuex)
export default new vuex.store({
  //...code
})

//main.js
import store from './store'
...
```

#### 7、**mvvm** 框架是什么？它和其他框架(**jquery**)的区别是什么？哪些场景适合

**mvvm** 是 **model + view + viewmodel** 框架，通过 **viewmodel** 连接数据模型 **model** 和 **view**

区别：**vue** 是数据驱动，通过数据来显示视图层而不是节点操用

场景：数据操作比较多的场景，更加快捷

#### 8、自定义指令(**v-check**, **v-focus**)的方法有哪些？它有哪些钩子函数？还有哪些钩子函数参数

全局定义指令：在 **vue** 对象的 **directive** 方法里面有两个参数，一个是指令名称，另一个是函数。

组件内定义指令：**directives**

钩子函数：**bind**(绑定事件出发)、**inserted**(节点插入时候触发)、**update**(组件内相关更新)

钩子函数参数：**el**、**binding**

## 9、说出至少 4 种 **vue** 当中的指令和它的用法

**v-if**(判断是否隐藏)、**v-for**(把数据遍历出来)、**v-bind**(绑定属性)、**v-model**(实现双向绑定)

## 10、**vue-router** 是什么?它有哪些组件

**vue-router** 是 **vue** 的路由插件，

组件：**router-link** **router-view**

## 11、**vue** 的双向绑定的原理是什么

**vue.js** 是采用数据劫持结合发布者-订阅者模式的方式，通过 **Object.defineProperty()** 来劫持各个属性的 **setter**，**getter**，在数据变动时发布消息给订阅者，触发相应的监听回调。

具体步骤：

第一步：需要 **observe** 的数据对象进行递归遍历，包括子属性对象的属性，都加上 **setter** 和 **getter** 这样的话，给这个对象的某个值赋值，就会触发 **setter**，那么就能监听到了数据变化

第二步：**compile** 解析模板指令，将模板中的变量替换成数据，然后初始化渲染页面视图，并将每个指令对应的节点绑定更新函数，添加监听数据的订阅者，一旦数据有变动，收到通知，更新视图

第三步：**Watcher** 订阅者是 **Observer** 和 **Compile** 之间通信的桥梁，主要做的事情是：

1、在自身实例化时往属性订阅器(**dep**)里面添加自己

2、自身必须有一个 **update()** 方法

3、待属性变动`dep.notice()`通知时，能调用自身的 `update()` 方法，并触发`Compile`中绑定的回调，则功成身退。

第四步：MVVM作为数据绑定的入口，整合`Observer`、`Compile`和`Watcher`三者，通过`Observer`来监听自己的`model`数据变化，通过`Compile`来解析编译模板指令，最终利用`Watcher`搭起`Observer`和`Compile`之间的通信桥梁，达到数据变化 -> 视图更新；视图交互变化(input) -> 数据`model`变更的双向绑定效果。

## 12、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

## vuex 面试题

### 1、有哪几种属性

有 5 种，分别是 `state`、`getter`、`mutation`、`action`、`module`

### 2、vuex 的 store 特性是什么

(1) **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 **data**

(2) **state** 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

(3) 它通过 **mapState** 把全局的 **state** 和 **getters** 映射到当前组件的 **computed** 计算属性

### 3、**vuex** 的 **getter** 特性是什么

(1) **getter** 可以对 **state** 进行计算操作，它就是 **store** 的计算属性

(2) 虽然在组件内也可以做计算属性，但是 **getters** 可以在多给件之间复用

(3) 如果一个状态只在一个组件内使用，是可以不用 **getters**

### 4、**vuex** 的 **mutation** 特性是什么

**action** 类似于 **muation**, 不同在于：**action** 提交的是 **mutation**, 而不是直接变更状态

**action** 可以包含任意异步操作

### 5、**vue** 中 **ajax** 请求代码应该写在组件的 **methods** 中还是 **vuex** 的 **action** 中

如果请求来的数据不是要被其他组件公用，仅仅在请求的组件内使用，就不需要放入 **vuex** 的 **state** 里

如果被其他地方复用，请将请求放入 **action** 里，方便复用，并包装成 **promise** 返回

### 5、不用 **vuex** 会带来什么问题

可维护性会下降，你要修改数据，你得维护3个地方

可读性下降，因为一个组件里的数据，你根本就看不出来是从哪里来的

增加耦合，大量的上传派发，会让耦合性大大的增加，本来**Vue**用**Component**就是为了减少耦合，现在这么用，和组件化的初衷相背

# 生命周期面试题

## 1、什么是 **vue** 生命周期

**vue** 实例从创建到销毁的过程就是生命周期。

也就是从开始创建、初始化数据、编译模板、挂在 **dom** -> 渲染、更新 -> 渲染、写在等一系列过程

## 2、**vue**生命周期的作用是什么

生命周期中有多个事件钩子，让我们在控制整个 **vue** 实例的过程时更容易形成好的逻辑

## 3、**vue**生命周期总共有几个阶段

8个阶段：创建前/后、载入前/后、更新前/后、销毁前/后

## 4、第一次页面加载会触发哪几个钩子

第一次加载会触发 **beforeCreate**、**created**、**beforeMount**、**mounted**

## 5、**DOM** 渲染在哪个周期中就已经完成

**mounted**

## 6、简述每个周期具体适合哪些场景

生命周期钩子的一些使用方法：

**beforecreate** : 可以在这加个**loading**事件，在加载实例时触发

**created** : 初始化完成时的事件写在这里，如在这结束**loading**事件，异步请求也适宜在这里调用

**mounted** : 挂载元素，获取到**DOM**节点 **updated** : 如果对数据统一处理，在这里写上相应函数

**beforeDestroy** : 可以做一个确认停止事件的确认框 **nextTick** : 更新数据后立即操作**dom**