面试题整理Vue：

## 面试题一、请你简述vue的生命周期钩子函数

beforeCreate 在实例初始化之后，数据观测(data observer) 和 event/watcher 事件配置之前被调用。

created 实例已经创建完成之后被调用。在这一步，实例已完成以下的配置：数据观测(data observer)，属性和方法的运算， watch/event 事件回调。这里没有$el。

beforeMount 在挂载开始之前被调用：相关的 render 函数首次被调用。

mounted el 被新创建的 vm.$el 替换，并挂载到实例上去之后调用该钩子。

beforeUpdate 数据更新时调用，发生在虚拟 DOM 重新渲染和打补丁之前。

updated 由于数据更改导致的虚拟 DOM 重新渲染和打补丁，在这之后会调用该钩子。

beforeDestroy 实例销毁之前调用。在这一步，实例仍然完全可用。

destroyed Vue 实例销毁后调用。调用后，Vue 实例指示的所有东西都会解绑定，所有的事件监听器会被移除，所有的子实例也会被销毁。 该钩子在服务器端渲染期间不被调用。

**钩子函数中该干的事情：**

created 实例已经创建完成，因为它是最早触发的原因可以进行一些数据，资源的请求。

mounted 实例已经挂载完成，可以进行一些DOM操作。

beforeUpdate 可以在这个钩子中进一步地更改状态，这不会触发附加的重渲染过程。

updated 可以执行依赖于 DOM 的操作。然而在大多数情况下，你应该避免在此期间更改状态，因为这可能会导致更新无限循环。 该钩子在服务器端渲染期间不被调用。

destroyed 可以执行一些优化操作,清空定时器，解除绑定事件。

## 面试题二、vue组件间通信常用方式

首先明确一点，组件化的好处：方便维护，易于扩展、减少DOM操作。

1. 使用props传递参数（属性在组件标签上需要使用短横线命名法，在组件中声明需要采用驼峰命名法；通过带有校验，通过type指明外界传递过来的数据是什么类型，其中，对象和数组类型如果给出默认值时需要返回一个工厂函数）
2. 子组件中使用$emit触发父组件中的方法，通过回调的方式讲修改的内容传递给父组件
3. .sync的形式，触发的事件名 update:(绑定.sync属性的名字) --属于固定写法

<Son1 :mny.sync="mny"></Son1>

<button @click="$emit('update:mny',200)">更改</button>

1. v-model，子组件触发的事件名称只能是input，接受到的属性名只能叫value
2. 使用$parent、$children，可以获取到父组件或者子组件中的属性和方法。如果层级很深，就会出现$parent.$parent，可以封装一个$dispatch方法向上进行派发，或者封装一个$broadcast向下派发
3. $attrs、$listeners 使用$attrs属性，可以将属性继续向下传递；使用$listeners将方法继续向下传递
4. Provide和Inject，向任意子组件中可以注入父级数据，使用inject接收到值以后，会将数据挂载在当前实例上
5. 使用ref 可以获取组件的实例，实例上的属性和方法随机可以获取到
6. 使用eventBus进行跨组件通信

## 面试题三、简单聊一下Promise

1、可以解决并发问题（可以同步多个异步方法的执行结果）

2、可以解决多个回调嵌套的问题

3、Promise是一个类；

4、每次可以通过new的形式实例化一个实例时都需要传递一个执行器（执行器是立即执行的），执行器函数中有两个参数（resolve、reject，这两个都是方法）；

5、默认Promise有三个状态，pendding（请求中），如果调用了resolve方法，表示将请求中的状态转到成功态（fulfilled），如果调用了reject方法，表示将请求中的状态转到失败态（rejected）；

6、只有状态是pendding态时，才能更改状态；并且，状态一旦改变，不能更改；

7、每一个promise实例都有一个then方法，then方法里面包含成功的回调和失败的回调

Promise.all 处理多个异步的并发问题（全部完成才算完成，如果有一个失败，就失败）

Promise.race 有一个成功就成功，有一个失败就失败

## 面试题四、说一下vue中的slot插槽

使用customer组件时，如果没有包含一个<slot>元素，则该组件起始标签和结束标签之间的任何内容都会被抛弃。

直接使用slot元素时也称为默认插槽，如果组件中需要放置多个位置，可以给slot添加name属性，用来定义额外的插槽。

指的一提的是：Vue2.6.0起引入了v-slot，取代了slot和slot-scope这两个目前已被废弃但未被移除且仍在文档中的attribute。

子节点数据会存在组件实例中的$slots.default

## 面试题五、vue中的render函数

Vue选项中的render函数若存在，则Vue构造函数不会从template选项或通过el选项指定的挂载元素中提取出的html模板编译渲染函数。该渲染函数接受一个createElement方法作为第一个参数用来创建 VNode。

虚拟DOM：不是一个实际的DOM元素。它更准确的名字可能是createNodeDescription，因为它所包含的信息会告诉Vue页面上需要渲染什么样的节点，包括及其子节点的描述信息。

createElement参数：一个标签名称（必填项），一个与模板中属性对应的数据对象（可选），子级虚拟节点，由createElement构建而成。也可以使用字符串来生成”文本虚拟节点“（可选）

## 面试题六、简述一下vuex

vuex是专为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. 你不能直接改变 store 中的状态。改变 store 中的状态的唯一途径就是显式地****提交 (commit) mutation****。
3. 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
4. getter就像计算属性一样，getter的返回值会根据它的依赖被缓存起来，且只有它的依赖值发生了变化才会被重新计算

## 面试题七、vue-router有关

Vue-router是vue官方的路由管理器。

提供了诸如：

hash和history两种路由模式。

使用该插件时，必须提供一个install方法，并且以Vue作为参数。

提供了两个全局组件router-link、router-view。

每个组件中都有$router和route 这两个属性，前者里面有很多api，后面里面有很多属性

如何实现一个vue-router。后面有代码示例，可以研读。

## 面试题八、vue中数组是如何监听变化的

vue源码中重写了关于操作数组的七个方法（push、pop、unshift、shift、reverse、sort、splice）

关键代码：

let oldProtoMehtods = Array.prototype;

let proto = Object.create(oldProtoMehtods);

['push', 'pop', 'shift', 'unshift','reverse','sort','splice' ].forEach(method => {  
 proto[method] = (...args) => {  
 oldProtoMehtods[method].call(this, ...args)

}  
})

另外，push，unshift，splice这三种添加的操作，还遍历了数组里面每一个元素，如果是对象时，则再次Object.defineProperty添加getter和setter一次。

## 面试题九、什么是库？什么是框架？什么是声明式？什么是命令式？

1、库是将代码合成一个产品，库是我们调用库里的方法实现自己的功能

2、框架则是为解决一类问题而开发的产品，框架是我们在指定的位置编写好代码，框架帮我们调用

**框架和库的最本质的区别在于控制权：you call libs, frameworks call you**

3、声明式就是利用数组的方法forEach（我们想要的是循环，内部帮我们去做）

4、自己写for循环就是命令式（命令其按照自己的方式得到结果）

## 面试题十、watch和computed的区别

watch和computed 内部调用的都是new Watcher

watch:

let vm = new Vue({

el: '#app',

data: { name: 'heima' }

});

function initWatch(key,handler){

vm.$watch(key,handler);

}

initWatch('name',function(newValue){

console.log(newValue)

});

vm.name = 'care'; // 数据变化会执行对应的handler

Computed:

let dirty = true; // 内部有脏值检测系统

function initComputed(key,handler) {

Object.defineProperty(vm, key, {

get() {

if (dirty) { // 取值后会将dirty 变成false

value = handler();

dirty = false;

}

return value

}

})

}

initComputed('age', function () {

console.log('取值一次')

return vm.name

});

console.log(vm.age);

console.log(vm.age);

vm.name = 'hello'; // 当依赖的值变化 会将dirty重新改成true,保证获取最新值

dirty = true;

console.log(vm.age);

## 面试题十一、MVC和MVVM的区别

M：表示数据模型层、V：表示视图层（html页面）、C：控制器

M：表示数据模型层、V：表示视图层（html页面）、VM：view-model表示核心调度者、协调器

在传统的mvc中除了model和view以外的逻辑都放在了controller中，导致controller逻辑复杂难以维护

在mvvm中view和model没有直接的关系，全部通过viewModel进行交互

## 面试题十二、在Vue生命周期的created()钩子函数进行的DOM操作一定要放在Vue.nextTick() 回调函数中。原因是什么呢

第一点，created钩子函数里是获取不到真实的DOM元素的

第二点，$nextTick方法，参数是一个回调函数。执行的时机是将该回调延迟到下次DOM更新循环之后执行。

nextTick的特点就是等待DOM更新

## 面试题十三、vuex中的store的特性是什么

store表示单一状态树，用一个对象就包含了全部的应用层级状态。它作为一个‘唯一数据源’而存在。单一状态树让我们能够直接地定位任一特定的状态片段，在调试过程中也能轻易地取得整个应用状态的快照。

## 面试题十四、vue和jquery的区别

vue是数据驱动视图（html页面），数据一变，页面中有关该数据的地方会自动刷新成最新

jquery是本着操作dom的初衷，先获取dom元素，再操作dom（增删改查）

## 面试题十五、数组怎么用v-model绑定

vue内部对数组的更新检测明确规定，只有七个变异方法（改变了调用这些方法的原始数组）

可以使用Vue.set(target, propertyName/index,value)，向响应式对象中或者数组中添加一个属性，并确保这个属性同样也是响应式的，且触发视图更新

## 面试题十六、data数据改变时，什么情况下页面数据不改变

无法检测到对象属性的添加或删除。由于Vue会在初始化实例时对属性进行getter/setter转化，所以属性必须在data对象上存在才能让Vue将它转成响应式。

当一个组件被定义，data必须声明为返回一个初始数据对象的函数，因为组件可能被用来创建多个实例。如果data仍然是一个纯粹的对象，则所有的实例将共享引用同一个数据对象！通过提供data函数，每次创建一个实例以后，我们能够调用data函数，从而返回初始数据的一个全新的副本数据对象

## 面试题十七、vue路由如何实现跳转

除了使用router-link创建a标签来定义导航链接，我们还可以借助router的实例方法，通过代码来实现

在vue实例内部，可以通过$router访问路由实例。因此可以调用this.$router.push，该方法的参数可以是一个字符串路径，或者一个描述地址的对象，例如：

字符串：

router.push(‘home’)

对象：

router.push({ path: ‘home’ })

## 面试题十八、axios请求和响应拦截器

通过npm install axios，或者通过script引入cdn链接地址即可获取一个叫axios的全局对象。直接使用axios下面的get和post方法传入对应请求的url地址即可发送一个请求，返回一个promise实例，通过then方法指明成功回调，在回调中获取响应结果即可。

响应拦截器：通过axios下面的interceptors属性可以添加拦截器，响应拦截 - response.use

需要传递一个函数，函数的参数默认是响应后的结果，最后必须return该结果，在这中间可以对响应结果作进一步处理。

## 面试题十九、vue双向绑定原理及其局限性

使用es5提供的api：Object.defineProperty 对对象中的所有属性进行添加getter、setter方法，并且是递归的形式添加getter、setter属性。

局限性：

不能对对象进行添加或者删除时具有响应式，必须使用Vue.set方法

数组进行操作索引的形式不具有响应式，必须使用Vue.set方法

## 面试题二十、vue-router的两种模式，分别有什么好处

具有hash、history两种模式，前者表现形式#/、后者表现形式为/

hash模式的好处：当浏览器路径变化时，页面不会刷新，用户体验比较好

history 模式的好处：浏览器路径不带有#，URL就像正常的url，不过这种模式要玩好，还需要后台配置支持。因为我们的应用是个单页客户端应用，如果后台没有正确的配置，很可能导致404的发生。所以呢，要在服务端增加一个覆盖所有情况的候选资源：如果URL匹配不到任何资源，则应该返回同一个index.html页面，这个页面就是你app依赖的页面。

## 面试题二十一、路由导航守卫

可以使用router.beforeEach注册一个全局前置守卫

每个守卫方法接收三个参数：

to：即将要进入的目标路由对象（你想跳到到哪个页面）

from：当前导航正要离开的路由（从哪个页面跳过来）

next：是一个方法，一定要调用该方法来resolve这个钩子。执行效果依赖next方法的调用参数。

next( )：进行管道中的下一个钩子

next(false)：中断当前的导航。如果浏览器的URL改变了，那么URL地址会重置到from路由对应的地址。

next(‘/’)或者next({ path: ‘/’ })：跳转到一个不同的地址。当前的导航被中断，然后进行一个新的导航。

## 面试题二十二：token过期怎么处理，怎么集中处理请求

真正企业中，发每个请求的时候，都要携带token给到后端，让后端判断此次请求是否有效，如果无效，前端自行判断去重新登录即可。

集中处理：可以在请求拦截器中进行统一处理，如果检测到token过期就让用户进行重新登录。

## 面试题二十三：封装过什么vue组件

常见日历组件，页面上有一个input输入框，当输入框获取焦点时弹出日历选择面板，用户在日历面板中进行选择具体日期，点击确定之后有两个动作，一个是面板消失；另一个是input输入框中的值就是刚刚用户选择的日期。

要点一：当输入框获取焦点时弹出日历面板，需要使用自定义指令的形式。否则，点击面板时会出现面板消失的情况。

要点二：就是对Date 函数的常用方法（获取年月日，设置年月日等等）

## 面试题二十四：vue-cli项目中src文件夹和文件的用法

assets文件夹是放静态资源

components 文件夹是专门放组件的

router 是定义路由相关的配置

views 是专门放页面的文件夹

app.vue 是一个应用组件（称之为根组件或者根实例即可）

main.js 是一个默认入口文件

## 面试题二十五：在 Vue 实例中编写生命周期 hook 或其他 option/propertie 时，为什么不使用箭头函数 ？

首先，箭头函数本身没有this，当你在用Vue程序中使用箭头函数时，this关键字不会绑定到Vue实例，因此会引发错误，比如会指向window全局，访问某个属性时很可能导致undefined的发生

Vue面试题整理第二部分

### 1、对vue的理解，有什么特点，vue为什么不能兼容IE8及以下浏览器

　　vue是一套用于构建用户界面的渐进式框架，核心是一个响应的数据绑定系统

　　vue是一款MVVM框架，基于双向绑定数据，当数据发生变化时候，vue自身会进行一些运算

　　特点：简洁轻量、数据驱动、组件化、模块友好

　　vue.js使用了IE8无法模拟的 ECMAScript 5 特性，没有替代方案

### 2、简述Vue双向数据绑定的原理

　Vue2.0版本中　主要是通过Object对象的defineProperty属性，重写data的set和get函数来实现的

　　vue是通过数据劫持的方式来做数据绑定，最核心的方法就是通过Object.defineProperty()来实现对属性的劫持

　　在设置或者获取的时候我们就可以在get或者set方法里加入其他的触发函数，达到监听数据变动的目的

 Vue3.0的一个主要点就是数据劫持的实现变更，其他的与Vue2.0大致相同。[Proxy](https://links.jianshu.com/go?to=http%3A%2F%2Fes6.ruanyifeng.com%2F%23docs%2Fproxy" \t "_blank) 是ES6中新增的一个特性，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。[Proxy](https://links.jianshu.com/go?to=http%3A%2F%2Fes6.ruanyifeng.com%2F%23docs%2Fproxy" \t "_blank) 这个词的原意是代理，用在这里表示由它来“代理”某些操作，可以译为“代理器”。

### 3、什么是MVVM，和MVC的区别

　　MVVM是Model-View-ViewModel的缩写

　　Model层代表数据模型

　　View代表组件视图，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（双向绑定）

　　在MVVM中，View和Model之间并没有直接的联系，而是通过ViewModel进行交互，

　　Model和ViewModel之间的交互是双向的，因此 通过视图操作数据，也能通过数据操作视图

　　MVC是Model-View- Controller的简写。即模型-视图-控制器，使用MVC的目的是为了将M和V相分离

　　MVVM与MVC最大的区别就是实现了View和Model的自动同步，也就是当Model的属性改变时

　　我们不用再手动操作Dom来改变View，而是改变后该属性对应View层会自动改变

### 4、vue.js的两个核心是什么

　　数据驱动和组件化思想

　　1、数据驱动，也叫双向数据绑定。

　　Vue.js数据观测原理在技术实现上，利用的是ES5Object.defineProperty和存储器属性: getter和setter（所以只兼容IE9及以上版本），可称为基于依赖收集的观测机制。核心是VM，即ViewModel，保证数据和视图的一致性。

　　2、组件系统。

　　vue组件的核心选项:

　　（1）模板（template）：模板声明了数据和最终展现给用户的DOM之间的映射关系。

　　（2）初始数据（data）：一个组件的初始数据状态。对于可复用的组件来说，这通常是私有的状态。

　　（3）接受的外部参数(props)：组件之间通过参数来进行数据的传递和共享。  
（4）方法（methods）：对数据的改动操作一般都在组件的方法内进行。  
（5）生命周期钩子函数（lifecycle hooks）：一个组件会触发多个生命周期钩子函数，最新2.0版本对于生命周期函数名称改动很大。  
（6）私有资源（assets）：Vue.js当中将用户自定义的指令、过滤器、组件等统称为资源。一个组件可以声明自己的私有资源。私有资源只有该组件和它的子组件可以调用。  
等等。

### 5、vue与angular的区别

　　vue的双向邦定是基于ES5中getter/setter来实现的，而angular是由自己实现一套模版编译规则，需要进行所谓的“脏值”检查，vue则不需要

　　vue需要提供一个el对象进行实例化，后续的所有作用范围也是在el对象之下，而angular而是整个html页面

### 6、说下vue的底层原理

　　Vue的模式是m-v-vm模式，即（model-view-modelView），通过modelView作为中间层，进行双向数据的绑定与变化

　　1）通过建立虚拟dom树document.createDocumentFragment(),方法创建虚拟dom树

　　2）一旦被监测的数据改变，会通过Object.defineProperty定义的数据拦截，截取到数据的变化

　　3）截取到的数据变化，从而通过订阅——发布者模式，触发Watcher（观察者）,从而改变虚拟dom的中的具体数据

　　4）最后通过更新虚拟dom的元素值，从而改变最后渲染dom树的值，完成双向绑定

### 7、简述vue等单页面应用及优缺点

　　单页面应用，用户所有的操作都在一个页面完成

　　优点：无刷新，用户体验好，共享资源只需要请求一次即可，采用组件化的思想，代码结构更加规范化，便于修改和调整

　　缺点：对搜索引擎不友好、低版本不支持，第一次加载首页耗时相对较长，不能使用浏览器导航按钮，需要自行实现前进后退

### 8、React与Vue对比

　　相同点：

　　都支持服务器端渲染、数据驱动视图，状态管理

　　都有虚拟DOM、组件化开发、通过props参数进行父子组件数据的传递

　　不同点：

　　React严格上只针对MVC的C层,Vue则是MVVM模式

　　虚拟DOM方面

　　vue会跟踪每一个组件的依赖关系,不需要重新渲染整个组件树，而React每当应用的状态被改变时,全部组件都会重新渲染

　　视图渲染方面

　　React采用JSX渲染到DOM，vue使用的是template模板

　　数据绑定方面

　　vue实现了数据的双向绑定，react数据流动是单向的

　　state对象方面

　　react应用中不可变的,需要使用setState方法更新状态

　　vue中,state对象不是必须的,数据由data属性在vue对象中管理

### 9、前端组件化有什么优势

　　1）提高开发效率

　　2）方便重复使用

　　3）便于协同开发

　　4）更容易管理维护

### 10、说一下vue的生命周期，当使用keep-alive属性时，会增加哪两个生命周期

　　创建前/后beforeCreate/created：

　　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beforeMount/mounted：

　　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尚未替换。在mounted阶段，vue实例挂载完成，data成功渲染。

　　更新前/后beforeUpdate/updated：

当data变化时，会触发beforeUpdate和updated方法。

数据更新，但是页面还没有更新时执行beforeUpdate，页面也更新完毕了 执行updated

　　销毁前/后beforeDestory/destoryed：

　　beforeDestory是在vue实例销毁前触发，一般在这里要通过removeEventListener解除手动绑定的事件

　　执行destroy方法后，vue实例已经解除了事件监听以及dom的绑定，但是dom结构依然存在

　　在被keep-alive包含的组件/路由中，会多出两个生命周期的钩子:

　　activated 与 deactivated

### 11、说说你对angular脏检查理解

angular中无法判断数据是否做了更改，所以设置了一些条件，当触发这些条件之后就会执行一个检测来遍历所有的数据，对比刚才更改的地方执行变化

AngularJS常用函数：$apply,$watch及$digest

$digest是一个内部函数，正常的应用代码中是不应该直接调用它的。要想主动触发它，就要调用scope.$apply函数，它是触发Angular“脏检查机制”的常用公开接口。$digest循环实际上包括两个while循环。分别是：处理$evalAsync的异步运算队列，处理$watch的watchers队列。当该循环触发时，它会遍历当前$scope及其所有子$scope上已注册的所有watchers函数。遍历一遍所有watcher函数成为一轮脏检查。执行完一轮脏检查，如果任何一个watcher所监听的值改变过，那么就会重新再进行一轮脏检查，直到所有的watcher函数都报告其所监听的值不改变了。当循环结束时，才把模型的变化结果更新到DOM中去。这样做为了，防止频繁更新DOM属性。

通俗理解脏检查：表达式{{aaa.x}},AngularJS不仅会渲染该数据，还会为该特定值创建一个观察程序。之后，只要程序发生任何事情，AngularJS就会检查该观察过程中的值是否更改。如果有，重新呈现表达式。运行这些观察者的过程称为脏检查

　　这个检查很不科学而且效率不高，有很多多余的地方

### 12、如何使css只在当前组件起作用

　　如果想写的css只对当前组件起作用，则在style中写入scoped

### 13、vue中v-if和v-show的区别

　　v-if和v-show都是用来控制元素的渲染

　　v-if是根据后面数据的真假，来判断DOM的添加删除等操作

　　v-show只是在修改元素的css样式（display属性值）

　　v-if如果初始渲染条件为真，就渲染，反之就不渲染

　　v-show不管初始条件是否为真，都会被渲染

　　v-if有更高的切换消耗，不适合做频繁的切换

　　 v-show有更高的初始渲染消耗，适合做频繁的切换

### 14、vue有哪些修饰符

　　事件修饰符：stop、prevent、self、once、native

　　键盘修饰符：enter、space、up、down

　　表单修饰符修饰符：number、trim、lazy

### 15、列举vue的几种常用指令

　　v-if、v-show、v-for、v-on、v-bind、v-model、v-once

### 16、v-on可以绑定多个方法吗？

　　可以

<input v-on:keyup.enter="submit" v-on:focus="onFocus">

### 17、vue中$mount与el区别

　　$mount和el两者在使用中没有什么区别，都是将实例化后的vue挂载到指定的DOM元素中  
　　如果在实例化vue时指定el，则该vue将会渲染在el对应的DOM中  
　　没有指定el，则vue实例会处于一种“未挂载”的状态，此时通过$mount来手动执行挂载

**18、vue事件修饰符怎么使用，举例说明**[IMG_257](https://www.cnblogs.com/theblogs/p/javascript:void(0);)

* 在事件处理程序中调用 event.preventDefault() 或 event.stopPropagation() 是非常常见的需求。
* Vue 不推荐我们操作DOM 为了解决这个问题，Vue.js 为 v-on 提供了**事件修饰符**
* 修饰符是由点开头的指令后缀来表示的

<!-- 阻止单击事件继续传播 -->

<a v-on:click.stop="doThis"></a>

<!-- 提交事件不再重载页面 -->

<form v-on:submit.prevent="onSubmit"></form>

<!-- 修饰符可以串联 即阻止冒泡也阻止默认事件 -->

<a v-on:click.stop.prevent="doThat"></a>

<!-- 只当在 event.target 是当前元素自身时触发处理函数 -->

<!-- 即事件不是从内部元素触发的 -->

<div v-on:click.self="doThat">...</div>

使用修饰符时，顺序很重要；相应的代码会以同样的顺序产生。因此，用 v-on:click.prevent.self 会阻止所有的点击，而 v-on:click.self.prevent 只会阻止对元素自身的点击。

**19、vue中key值和is的作用**

使用key，vue会基于key的变化重新排列元素顺序，并且移除key不存在的元素。可以做优化处理

key的作用主要是为了高效的更新虚拟DOM。另外vue中在使用相同标签名元素的过渡切换时，也会使用到key属性，其目的也是为了让vue可以区分它们，否则vue只会替换其内部属性而不会触发过渡效果

　　is可以扩展原生html元素，也可以绑定动态组件

**20、组件中data为什么是函数**

　　因为组件是用来复用的，JS 里对象是引用关系，这样作用域没有隔离

　　而 new Vue 的实例，是不会被复用的，因此不存在引用对象的问题

**21、什么是单向数据流动**

　　这个概念出现在组件通信。数据从父级流向子级，数据本身还是父级的

　　如果操作子级要改变父级的数据，只能通过子级告知父级要操作哪个数据

　　然后让父级去修改自己的数据，修改完毕再传给子级

**22、什么是vue的计算属性，有几种用法**

　　computed 计算属性，能监听vue中数据的变化，当数据发生变化时候会触发

　　1、函数形式

computed:{

listenArr(){

//使用data中数据，自动帮你监听数据的变化

//返回的结果就是通过改变数据 做的另一件事情 }

}

　　2、对象形式

[IMG_258](https://www.cnblogs.com/theblogs/p/javascript:void(0);)

computed:{

listenArr:{

get(){

//获取时候 },

set(newVal){

//修改时候 }

}

}

//当使用get set时候，computed中定义的属性为一个对象

//不使用get set时候，computed中定义的属性可以是一个函数

[IMG_259](https://www.cnblogs.com/theblogs/p/javascript:void(0);)

**23、说一下vue组件之间的传递**

　　父级传递子级：

　　首先在子组件上加一个v-bind:自定义属性,等于父级的数据，子组件通过props方法接收数据

　　子级传递父级：

　　在子组件上绑定一个自定义事件，并且传入父级的“事件”处理函数

　　在子组件内部使用$emit监听这个自定义事件，this.$emit('自定义事件名',参数)

 使用ref获取子组件实例

使用$parent获取父组件实例

使用provide和inject广播数据

**24、vue中computed和watch的区别**

　　computed能够监听vue数据上的变化，页面上来就执行一次，每改变一次数据就又触发，在操作数据的时候，会派生出另一个事情

　　watch当指定数据发生变化时候触发。一开始不会触发，只有指定的数据发生变化就又触发一次

　　watch可以deep深度添加，computed不可以

**25、对vue中nextTick的理解**

**vue是依靠数据驱动视图更新的，该更新的过程是异步的**

**Vue 实现的响应式并不是数据发生变化之后视图立即变化**

**如果数据更新之后，立刻操作元素，是无法获取到对应数据更新之后的元素的，因为vue的视图更新时异步的，所以nextTikck可以理解为等视图更新完毕之后再执行其中的回调。在nextTick的回调中操作dom**

　　vue中nextTick可以拿到更新后的DOM元素

　　如果在mounted下不能准确拿到DOM元素，可以使用nextTick

　　在Vue生命周期的created()钩子函数进行的DOM操作一定要放在Vue.nextTick()的回调函数中

**26、自定义指令(v-check,v-focus) 方法有哪些? 它有哪些钩子函数? 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,分别为指令名称、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)

一个指令定义对象可以提供如下几个钩子函数 (均为可选)：

* 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。
* inserted：被绑定元素插入父节点时调用 (仅保证父节点存在，但不一定已被插入文档中)。
* update：所在组件的 VNode 更新时调用，**但是可能发生在其子 VNode 更新之前**。指令的值可能发生了改变，也可能没有。但是你可以通过比较更新前后的值来忽略不必要的模板更新 (详细的钩子函数参数见下)

钩子函数参数： el、binding

指令钩子函数会被传入以下参数：

* el：指令所绑定的元素，可以用来直接操作 DOM。
* binding：一个对象，包含以下属性：
  + name：指令名，不包括 v- 前缀。
  + value：指令的绑定值，例如：v-my-directive="1 + 1" 中，绑定值为 2。
  + oldValue：指令绑定的前一个值，仅在 update 和 componentUpdated 钩子中可用。无论值是否改变都可用。
  + expression：字符串形式的指令表达式。例如 v-my-directive="1 + 1" 中，表达式为 "1 + 1"。
  + arg：传给指令的参数，可选。例如 v-my-directive:foo 中，参数为 "foo"。
  + modifiers：一个包含修饰符的对象。例如：v-my-directive.foo.bar 中，修饰符对象为 { foo: true, bar: true }。
* vnode：Vue 编译生成的虚拟节点。
* oldVnode：上一个虚拟节点，仅在 update 和 componentUpdated 钩子中可用。

**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，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方式引入，在调用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

**28、请分别说出vue修饰符trim、number、lazy的作用**

　　trim:用来过滤前后的空格

　　number:将用户输入的数据绑定为number类型

　　lazy:使用了这个修饰符将会从“input事件”变成change事件进行同步

**29、vue-router 有哪几种导航钩子，它们有哪些参数**

　　全局导航钩子：to、from、next

　　组件内的钩子：beforeRouteEnter、beforeRouteUpdate、beforeRouteLeave

　　单个路由独享的钩子：beforeEnter

**30、路由之间如何进行跳转**

　　声明式（标签跳转） <router-link :to="xxx">

　　编程式（ js 跳转） router.push('index')

### 31、说一下懒加载（按需加载路由）

　　webpack 中提供了 require.ensure()来实现按需加载。以前引入路由是通过 import 这样的方式引入，改为 const 定义的方式进行引入

　　不进行页面按需加载引入方式

import home from '../common/home.vue'

　　进行页面按需加载的引入方式：

const home = r => require.ensure( [], () => r (require('../common/home.vue')))

### 32、vue中$router和$route区别

　　router是VueRouter的一个对象，通过Vue.use(VueRouter)，和VueRouter构造函数得到一个router的实例对象

　　route是一个跳转的路由对象，每一个路由都会有一个route对象

　　是一个局部的对象，可以获取对应的name,path,params,query等

### 33、vue-loader是什么？使用它的用途有哪些

## 在webpack中配置.vue组件页面的解析

　　vue文件的一个加载器，跟template/js/style转换成js模块

　　用途：js可以写es6、style样式可以scss或less、template可以加jade等

### 34、vue-router中query与params区别

　　 query和params两者都是在Vue路由中传参

　　query用path来引入，params只能用name来传递，不能使用path

query更像get请求（地址栏会显示参数），而params更像post（不会在地址栏显示参数）

query方式生成的url为/xx?id=id，params方式生成的url为xx/id

**params方式需要注意的是需要定义路由信息如：path: '/xx/:id',这样才能进行携带参数跳转，否则url不会进行变化，并且再次刷新页面后参数会读取不到**

### 35、怎么定义vue-router的动态路由？怎么获取传过来的动态参数

　　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

### 36、嵌套路由怎么定义

　　在 VueRouter 的参数中使用 children 配置，这样就可以很好的实现路由嵌套

　　children 里面是子路由，当然子路由里面还可以继续嵌套子路由

### 37、active-class是哪个组件的属性？

　　vue-router模块的router-link组件

### 38、举例vue常用的路由对象，以及作用

　　$route.name 当前路由的名称

　　$route.path 当前路由对象的路径

　　$route.meta 在路由里面埋一个字段，当切换路由时候把信息传过去

　　$route.query 查询信息包含路由中查询参数的键值对

　　$route.hash 当前路径的哈希值，带#

　　$route.params 预设的变量，切换时候通过parmas带过去某个id的值

### 39、请说出vue.cli项目中src目录每个文件夹和文件的用法

　　main.js是入口文件

　　app.vue是一个主组件

　　view放视图页面

　　components放组件

　　router是定义路由相关的配置

　　assets文件夹是放静态资源

### 40、vue.cli中怎样使用自定义的组件

　　第一步：在components目录新建你的组件文件

　　第二步：在需要用的页面（组件）中导入

　　第三步：注入到vue的子组件的components属性上面

　　第四步：在template视图view中使用

### 41、什么是vue生命周期，作用是什么，第一次页面加载会触发哪几个钩子

　　vue实例从创建到销毁的过程

　　也就是从开始创建、初始化数据、编译模板、挂载Dom→渲染、更新→渲染、卸载等一系列过程，我们称这是 Vue 的生命周期

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

　　第一次页面加载时会触发 beforeCreate, created, beforeMount, mounted 这几个钩子

### 

### 42、简单描述每个周期具体适合哪些场景

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

　　created : 初始化完成时的操作，比如结束loading事件，也可以操作异步请求

　　mounted : 挂载元素，获取到DOM节点，推荐使用nextTick

　　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，关掉定时器

　　destroyed：当前组件已被删除，清空相关内容

### 43、说一下vue中slot插槽

　　插槽就是vue实现的一套内容分发的API，将插槽元素作为承载分发内容的出口。

　　在组件模板中默认占一个位置，当使用组件标签时候，组件标签的内容会自动替换掉内容

　　slot中可以设置一些默认的内容，如果传递了内容则会替换掉，如果没有名字的标签会默认放到default中

### 44、computed、watch各是一个对象时，它们有哪些选项？ computed 和 methods 有什么区别？ computed 是否能依赖其它组件的数据？

　　computed有get set两个选项

　　watch有handler deep 是否深度，immeditate 是否立即执行

　　methods是一个对象，可以在对象中定义一个个方法，能接受参数，而computed不能

　　computed是可以缓存的，methods不会

　　computed可以依赖其他computed，甚至是其他组件的data

### 45、如何理解Vue中的Render渲染函数

　　Vue一般使用template来创建HTML，然后有的时候，我们需要使用javascript来创建html，这时候我们需要使用render函数

　　render函数return一个createElement组件中的子元素存储在组件实列中

　　createElement返回的是包含的信息会告诉VUE页面上需要渲染什么样的节点。我们称这样的节点为虚拟DOM

### 46、vuex是什么？怎么使用？哪种功能场景使用它？

　　vuex是vue框架中的状态管理器

　　在main.js引入store，注入，新建了一个目录store，export导出

　　适用场景：单页应用中，组件之间的状态。音乐播放、登录状态、加入购物车

### 47、在Vue生命周期的created()钩子函数进行的DOM操作一定要放在Vue.nextTick() 回调函数中。原因是什么呢

　　在 created() 钩子函数执行的时候DOM其实并未进行任何渲染，而此时进行 DOM 操作无异于徒劳，

　　所以在数据变化后要执行的某个操作，而这个操作需要使用随数据改变而改变的DOM结构的时候，这个操作都应该放进Vue.nextTick() 的回调函数中

### 48、vuex中store的特性是什么

　　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 data

　　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改变

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

### 49、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