1. **对vue响应式的了解。**

答：了解，数据改变，UI页面做出响应。当修改 Vue 实例中的 data 属性时，UI页面中的 data 会做出响应，Vue 是通过Object.defineProperty来实现数据响应的。const vm = new Vue ({data:myDate})。

1. **对vue3了解吗？**

答：自己看文档了解一些，Vue3.0又被称为vue-next,主要有3个特点: 响应式、模板、对象式的组件声明方式，进行了全面的更改，底层实现和上层的api都有了明显的变化，基于es6的Proxy重新实现了响应式，基于treeshaking内置了更多的功能，提供了类式的组件声明方式。

1. **vue生命周期。（\*\*\*）**

答：分为三个阶段，创建期、运行期、销毁期。

创建期：

1. beforeCreate：实例刚在内存中被创建出来，此时，还没有初始化好 data 和 methods 属性
2. create：实例已经在内存中创建OK，此时 data 和 methods 已经创建OK，此时还没有开始 编译模板
3. beforeMount：此时已经完成了模板的编译，但是还没有挂载到页面中
4. mounted：此时，已经将编译好的模板，挂载到了页面指定的容器中显示

运行期：

1.beforeUpdate：状态更新之前执行此函数， 此时 data 中的状态值是最新的，但是界面上显示的 数据还是旧的，因 为此时还没有开始重新渲染DOM节点。

2. updated：实例更新完毕之后调用此函数，此时 data 中的状态值 和 界面上显示的数据，都已经完成了更新，界面 已经被重新渲染好了！

销毁期：

1. beforeDestroy：实例销毁之前调用。在这一步，实例仍然完全可用。

2. 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。

1. **computed和watch的原理知道吗,computed和watch区别。**

答：原理：computed 内部实现了一个惰性的 watcher,也就是 \_computedWatchers，\_computedWatchers 不会立刻求值，同时持有一个 dep 实例。Watch：vm 调用 $watch 后，首先调用 observe 函数 创建 Observer 实例观察数据，Observer 又创建 Dep , Dep 用来维护订阅者。然后创建 Watcher 实例提供 update函数。一旦数据变动，就层层执行回调函数。

区别：

computed 计算属性 : 依赖其它属性值，并且 computed 的值有缓存，只有它依赖的属性值发生改变，下一次获取 computed 的值时才会重新计算 computed 的值。

watch 侦听器 : 更多的是「观察」的作用，无缓存性，类似于某些数据的监听回调，每当监听的数据变化时都会执行回调进行后续操作。其内部通过 this.dirty 属性标记计算属性是否需要重新求值

1. **vue路由有哪几种形式。**

答：三种，router-link （声明式路由）、router.push(编程式路由)、this.$router.push() (函数里面调用)

1. **路由传参的方法。**

答：第一种this.$router.push( { path: `/particulars/${id}`,} )，需要对应路由配置一下：{ path: '/particulars/:id', name: 'particulars', component: particulars }。第二种 this.$router.push( { path: `/particulars/${id}`, params: { id: id } } )对应路由不需要在配置:/id了。第三种this.$router.push( { path: `/particulars/${id}`, query: { id: id } } )对应路由也不需要在配置:/id了。

1. **路由权限控制。**

答：两种方法，第一种vue-router 在构建路由时提供了元信息 meta 配置接口，我们可以在元信息中添加路由对应的权限，然后在路由守卫中检查相关权限，控制其路由跳转。可以在每一个路由的 meta 属性里，将能访问该路由的角色添加到 roles 里。用户每次登陆后，将用户的角色返回。然后在访问页面时，把路由的 meta 属性和用户的角色进行对比，如果用户的角色在路由的 roles 里，那就是能访问，如果不在就拒绝访问。第二种一般的，用户登录后会在本地存储用户的认证信息，可以用 token 、 cookie 等,这里我们用 token 。将用户的 token 保存到 localStorage 里，而用户信息则存在内存 store 中。这样可以在 vuex 中存储一个标记用户登录状态的属性 auth ，方便权限控制。

1. **知道异步组件吗？怎么用？**

答：使用过，Vue.component异步组件这个函数它接受一个resolve回调，这个回调函数会在你从服务器得到组件定义的时候被调用。你也可以调用 reject(reason) 来表示加载失败。这里的 setTimeout 是为了演示用的，如何获取组件取决于你自己。一个推荐的做法是将异步组件和 webpack 的 code-splitting 功能一起配合使用，你也可以返回一个Promise，() => import('组件路径')。

1. **虚拟DOM原理？**

答：用js对象构造一个虚拟的dom树，插入到文档中，它其实就是一个对象，当状态变更时，使用diff算法去计算新树和旧树的差异，把上面的差异构建到真正的dom中。

1. **组件销毁后，定时器还存在吗**

答：并没有，依然执行。我们只有在beforeDestroy钩子函数中去销毁clearTimeout

1. **你这个项目用到了vant里的哪些组件。**

答：Mint UI，iview，antd，element 基本的都有去使用过。

1. **你们vant是全局引入还是按需引入？**

答：按需引入的，可以达到减小项目体积的目的。

1. **双向数据绑定怎么实现的？（\*\*\*）**

答：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。就必须要实现以下几点： 1、实现一个数据监听器Observer，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，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 。

1. **有用代理（面试官是指proxy）写过什么代码吗？**

答：使用过proxy通过get设置代理对象的属性，在get中做判断return出是经过代理的对象，而不是原对象

1. **你vue用的技术栈有哪些？**

答：Vue+Vue-router+Vuex+axios+webpack+webpack-cli

1. **Vue自定义插件**

答：插件通常会为 Vue 添加全局功能，一般是添加全局方法/全局指令/过滤器等，通过全局方法 Vue.use() 使用插件，它需要在你调用 new Vue() 启动应用之前完成。

1. **介绍vuex**

答：VueX是适用于在Vue项目开发时使用的状态管理工具。试想一下，如果在一个项目开发中频繁的使用组件传参的方式来同步data中的值，一旦项目变得很庞大，管理和维护这些值将是相当棘手的工作。为此，Vue为这些被多个组件频繁使用的值提供了一个统一管理的工具——VueX。State：是一个单一的状态树，用来存储状态的。Geeter：可以认为是 store 的计算属性，getter 的返回值会根据它的依赖被缓存起来，且只有当它的依赖值发生了改变才会被重新计算。Mutation：更改 Vuex 的 store 中的状态的唯一方法是提交 mutation。Vuex 中的 mutation 非常类似于事件：每个 mutation 都有一个字符串的 事件类型 (type) 和 一个 回调函数 (handler)。这个回调函数就是我们实际进行状态更改的地方，并且它会接受 state 作为第一个参数。Action：类似于 mutatio 不同在于，Action 可以包含任意异步操作。Module： 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——从上至下进行同样方式的分割

1. **权限管理路由怎么实现？**

答：我们可以使用全局的路由守卫，前端定义好路由，并且在路由meta元数据上标记相应的权限信息。全局路由守卫每次都判断用户是否已经登录，没有登录则跳到登录页。已经登录(已经取得后台返回的用户的权限信息(角色之类的))，则判断当前要跳转的路由，用户是否有权限访问(根据路由名称到全部路由里找到对应的路由，判断用户是否具备路由上标注的权限信息(比如上面的roles: ['admin', 'editor']))。没有权限则跳到事先定义好的界面(403,404之类的)。这种方式，菜单可以直接用路由生成(用户没有权限的菜单也会显示，点击跳转的时候才做权限判断)，也可以在用户登录后根据用户权限把路由过滤一遍生成菜单(菜单需要保存在vuex里)。

1. **如何在vue的实现权限管理**

答：这个分成两块，第一接口访问的权限控制第二页面的权限控制，菜单中的页面是否能被访。接口权限就是对用户的校验。正常来说，在用户登录时服务器需要给前台返回一个Token,然后在以后前台每次调用接口时都需要带上这个Token, 然后服务端获取到这个Token后进行比对，如果通过则可以访问。页面权限的话就是先通过导航守卫来控制跳转路由

1. **做站点有一个登录的要求，登录的拦截，用vue怎么做**

答：可以通过在配置路由meta元数据上添加requireAuth属性，表明该路由是否需要登录验证，在进行全局拦截时，我们将通过该属性判断路由的跳转，在通过router中的钩子函数beforeEach结合requireAuth属性判断网页是否需要登录验证、再使用vuex进行token判断， 就能实现登录拦截的功能。

1. **这一个容器组件你如果给我一个值，就相应的输出这个值，如果不给我值我就显示默认值，如果出现这种情况一般怎么做？**

答：props中提供了type属性校验变量类型是否一致，default属性可以定义默认值。这样的话夫组件传递给子组件的时候，有值则返回传递的值，不传值则返回默认值。

1. **介绍一下vue常用的指令？（\*\*\*）**

答：1. v-html 简写 {{}} 绑定data中的值，放到标签的innerHTML。

2. v-text - v-html 基本一样， 但是它会把标签字符串原样输出。

3. v-bind 简写 ：:[] 标签的属性值（attribute）跟data中的数据进行绑定。

4. v-for 循环结构遍历 v-for循环的是它的载体以及所有子元素。

5. v-if 与 v-else - v-show v-if v-else 是对div节点加载与卸载（删除，添加），

而v-show 隐藏显示是控制style（样式）内的dispaly的属性 （none, block）。

6. v-model 一般用与input标签，作用就是绑定input的vue值 双向绑定。

7. v-on 简写 @ :[] 它用于监听 DOM 事件。

1. **v-for有什么作用？**

答：我们可以通过v-for指令来渲染一个数组列表，但在遍历渲染数据的时候，有一个属性key，但只能使用number和string，它回去做一个虚拟DOM的diff算法。

1. **vue路由懒加载实现**

答：结合 Vue 的[异步组件](https://cn.vuejs.org/v2/guide/components-dynamic-async.html" \l "%E5%BC%82%E6%AD%A5%E7%BB%84%E4%BB%B6" \t "_blank)和 Webpack 的[代码分割功能](https://doc.webpack-china.org/guides/code-splitting-async/" \l "require-ensure-/" \t "_blank)，实现路由组件的懒加载。首先，可以将异步组件定义为返回一个 Promise 的工厂函数 (该函数返回的 Promise 应该 resolve 组件本身)。其次，在 Webpack 2 中，我们可以使用[动态 import](https://github.com/tc39/proposal-dynamic-import" \t "_blank)语法来定义代码分块点 (split point)。

结合这两者，这就是如何定义一个能够被 Webpack 自动代码分割的异步组件。

1. **浏览器传参可以怎么做？**

答：（1）利用URL传参：在页面跳转的时候通过设置window.location.href添加参数，在接收参数的页面通过window.location.search获取参数字符串。（2）利用本地存储传参：可以使用本地存储的方式，可以使用cookie、sessionStorage和localStorage。

1. **对mvvm了解**

答：MVVM是Model-View-ViewModel的简写。它本质上就是MVC 的改进版。MVVM 就是将其中的View 的状态和行为抽象化，让我们将视图 UI 和业务逻辑分开。当然这些事 ViewModel 已经帮我们做了，它可以取出 Model 的数据同时帮忙处理 View 中由于需要展示内容而涉及的业务逻辑。

1. **Dom的渲染在哪个周期**

答：在mounted这个周期渲染DOM

1. **组件通讯是怎么做的（\*\*\*）**

答：• 父子通信：Props / $emit，$emit / $on，Vuex，$attrs / $listeners，provide/inject，$parent / $children＆$refs

• 兄弟通信：$emit / $on，Vuex

• 隔**代（跨级）通信：$emit / $on，Vuex，provide / inject，$attrs / $listeners**

1. **自定义指令**

答：1 注册：注册自定义指令分为全局注册与局部注册两种。

2 属性：自定义指令属性包含一些钩子函数，它们都是可选的。

3 多值入参：我们可以通过 JavaScript 的对象字面量，为自定义指令一次性传入多个值。其实任意一个合法的 JavaScript 表达式都是支持的。

1. **Key的作用，能不能用index作为key**

答：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。不能用index作为key

1. **Vue Router路有用过吗？**

答：Vue Router 是 [Vue.js](http://cn.vuejs.org/" \t "_blank) 官方的路由管理器。有hash模式和history模式。

包含的功能有：

* 嵌套的路由/视图表
* 模块化的、基于组件的路由配置
* 路由参数、查询、通配符
* 基于 Vue.js 过渡系统的视图过渡效果
* 细粒度的导航控制
* 带有自动激活的 CSS class 的链接
* HTML5 历史模式或 hash 模式，在 IE9 中自动降级
* 自定义的滚动条行为

1. **过滤器用过吗**

答：过滤器是对即将显示的数据做进一步的筛选处理，然后进行显示，值得注意的是过滤器并没有改变原来的数据，只是在原数据的基础上产生新的数据。

过滤器的使用：1、定义过滤器：分全局过滤器、局部过滤器

2、过滤器使用的地方：过滤器应该被添加在 JavaScript 表达式的尾部，由“管道”符号指示。

1. **MVVM和MVC的区别？**

答：MVC：在Controller（C）里面把Model（M）的数据赋值给View（V）。

MVVM：MVVM是Model-View-ViewModel的简写。它本质上就是MVC 的改进版。MVVM 就是将其中的View 的状态和行为抽象化，让我们将视图 UI 和业务逻辑分开。当然这些事 ViewModel 已经帮我们做了，它可以取出 Model 的数据同时帮忙处理 View 中由于需要展示内容而涉及的业务逻辑。

1. **Vue为什么采用异步渲染？**

答：如果不采取异步更新，那么每次更新数据都会对当前组件进行重新渲染，为了性能考虑，Vue 会在本轮数据更新后，再去异步更新数据，原理如下：

dep.notify() 通知 watcher 进行更新操作-->subs[i].update() 依次调用 watcher 的 update-->queueWatcher 将 watcher 重新放到队列中-->nextTick(flushSchedulerQueue) 异步清空 watcher 队列

1. **Vue model都是什么意思**

答：vue中经常使用到<input>和<textarea>这类表单元素，vue对于这些元素的数据绑定和我们以前经常用的jQuery有些区别。vue使用v-model实现这些标签数据的双向绑定，它会根据控件类型自动选取正确的方法来更新元素。

1. **说一下路由的钩子函数（路由守卫）？（\*\*\*）**

答：路由跳转前做一些验证，比如登录验证，是网站中的普遍需求。对此，vue-route 提供的beforeRouteUpdate可以方便地实现导航守卫，具体有：

* **全局前置守卫：router.beforeEach(to,from,next)**
* **全局解析守卫:router.beforeResolve**
* **全局后置钩子:router.afterEach(to,from)**
* **路由独享守卫:beforeEnter**
* **组件内的守卫:beforeRouteEnter()、beforeRouteUpdate()、beforeRouteLeave()？**
* Vue组件中常见的三种传值方式：父传子、子传父、非父子传值

1. **v-if 和 v-show有什么区别**

答：1.共同点 v-if 和 v-show 都可以动态地显示DOM元素 2.区别 (1)手段: v-if 是动态的向DOM树内添加或者删除DOM元素; v-show 是通过设置DOM元素的display样式属性控制显隐; (2)编译过程: v-if 切换有一个局部编译/卸载的过程

1. **Vue的父子，子父的通讯关系？（\*\*\*）**

答：一般情况下不管是组件还是实例，自己的数据只能自己使用，

如果子组件想使用父组件的数据，我们可以通过props来接收自定义属性的方式来将父组件的数据传递给子组件

具体流程如下：

1.在组件标签上使用自定义属性

2.在组件内部通过prop上来接收自定义属性

3.接收完毕以后，就可以使用父组件的数据

和父子通信不同的是，子父通信是通过子组件使用$emit去触发父组件的内置事件

1. **Vue里面是怎么操作dom的？**

答：**this.$refs.\* 拿到虚拟Dom,可以进行真实dom的一切操作**

**自定义指令中，el就是当前环境下的Dom元素**

var el = event.target; //当前元素，可修改（能够用此方法获取到他的子元素，不能获取他本身的内容）

var el = event.currentTarget;//当前元素，不可修改（能够用此方法获取到他的子元素及能获取他本身的内容）

1. **vue的里面的滚动是怎么操作的？**

答：创建一个 Router 实例，你可以提供一个 scrollBehavior 方法：

var router = new VueRouter({

  routes: [...],

  scrollBehavior (to, from, savedPosition) {

    // return 期望滚动到哪个的位置

  }

})

1. **Vue的2.0的脚手架 里面有两个文件夹有什么区别？是在什么环境下执行的？**

答：分别代表正式环境和测试环境，需要配合不同的启动命令启动

1. **vue-router有哪几种导航钩子**

答：（1）全局守卫：是所有路由配置的组件都会触发的，简单来说就是触发路由就会触发这些钩子函数，钩子函数的执行顺序为beforeEach、beforeResove、afterEach三个

（2）路由独享守卫：是指在单个路由配置的时候也可以设置的钩子函数，且只能在这个路由下起作用，目前只有一个钩子函数beforeEnter

（3）局部守卫：是指在组件内执行的钩子函数，类似与数组内的生命周期函数，相当于为配置路由的组件添加的生命周期钩子函数，钩子函数的执行顺序为beforeRouteEnter、beforeRouterUpdate、beforeRouterLeave三个

1. **防抖节流原理?**

答：**防抖： 在规定的时间内不触发第二次，则执行函数**

* + 1. 创建一个函数
    2. 函数内定义一个timer为null
    3. 借助闭包函数内返回一个函数
    4. 在内部调用的函数中判断如果timer有值就清除定时器，清除之后重新设置setTimeout，给定时器传入需要防抖函数与时间，
    5. 在内部调用的函数中判断如果timer没有值，直接设置timer为setTimeout,给定时器传入需要防抖的函数与时间

**节流： 当持续触发事件时，保证一定时间内只调用一次函数**

* + 1. 创建一个节流函数
    2. 函数内定义一个值为布尔的开关，开关为true
    3. 借助闭包函数内返回一个函数
    4. 在内部调用的函数做判断，如果定义的布尔值开关为false 直接返回false，否则开关为false，之后设置setTimeout，定时器中传入一个需要节流的函数并吧开关设置为true