**Vue**

1、Vue 的最大的优势是什么？（必会）

Vue 作为一款轻量级框架、简单易学、双向数据绑定、组件化、数据和结构的分离、虚拟

DOM、运行速度快，并且作者是中国人尤雨溪，对应的 API 文档对国内开发者优化，作为前端

开发人员的首选入门框架

**Vue 的优势：**

1、Vue.js 可以进行组件化开发，使代码编写量大大减少，读者更加易于理解。

2、Vue.js 最突出的优势在于可以对数据进行双向绑定。

3、使用 Vue.js 编写出来的界面效果本身就是响应式的，这使网页在各种设备上都能显示出非常好看的效果。

4、相比传统的页面通过超链接实现页面的切换和跳转，Vue 使用路由不会刷新页面。

5、vue 是单页面应用，使页面局部刷新，不用每次跳转页面都要请求所有数据和 dom，这样大大加快了访问速度和提升用户体验。

6、而且他的第三方 UI 组件库使用起来节省很多开发时间，从而提升开发效率。

2、Vue 和 jQuery 两者之间的区别是什么？（必会）

**1、jQuery 介绍：**

jQuery 曾经也是现在依然最流行的 web 前端 js 库，可是现在无论是国内还是国外他的使用率正在渐渐被其他的 js 库所代替，随着浏览器厂商对 HTML5 规范统一遵循以及ECMA6 在浏览器端的实现，jQuery 的使用率将会越来越低

**2、vue 介绍：**

vue 是一个兴起的前端 js 库，是一个精简的 MVVM。从技术角度讲，Vue.js 专注于 MVVM模型的 ViewModel 层。它通过双向数据绑定把 View 层和 Model 层连接了起来，通过对数据的操作就可以完成对页面视图的渲染。当然还有很多其他的 mvmm 框架如 Angular，react 都是大同小异，本质上都是基于 MVVM 的理念，然而 vue 以他独特的优势简单，快速，组合，紧凑，强大而迅速崛起

**3、vue 和 jQuery 区别：**

3.1）vue 和 jQuery 对比 jQuery 是使用选择器（）选取 DOM 对象，对其进行赋值、取值、事件绑定等操作，其实和原生的 HTML 的区别只在于可以更方便的选取和操作 DOM 对象，而数据和界面是在一起的

3.2）比如需要获取 label 标签的内容：）选取 DOM 对象，对其进行赋值、取值、事件绑定等操作，其实和原生的 HTML 的区别只在于可以更方便的选取和操作 DOM 对象，而数据和界面是在一起的

3.3）比如需要获取 label 标签的内容：(“lable”).val();,它还是依赖 DOM 元素的值。Vue 则是通过 Vue 对象将数据和 View 完全分离开来了

3.4）对数据进行操作不再需要引用相应的 DOM 对象，可以说数据和 View 是分离的， 他们通过 Vue 对象这个 vm 实现相互的绑定，这就是传说中的 MVVM

3、MVVM 和 MVC 区别是什么？哪些场景适合？（必会）

**1、基本定义**

1.1）MVVM 基本定义

MVVM 即 Model-View-ViewModel 的简写，即模型-视图-视图模型，模型（Model）指的是后端传递的数据，视图(View)指的是所看到的页面，视图模型(ViewModel)是 mvvm 模式的核心，它是连接 view 和 model 的桥梁。它有两个方向：

1.1.1）一是将模型（Model）转化成视图(View)，即将后端传递的数据转化成所看到 的页面，实现的方式是：数据绑定，

1.1.2）二是将视图(View)转化成模型(Model)，即将所看到的页面转化成后端的数据。实现的方式是：DOM 事件监听，这两个方向都实现的，我们称之为数据的双向绑定

1.2）MVC 基本定义

MVC 是 Model-View- Controller 的简写。即模型-视图-控制器。M 和 V 指的意思和MVVM 中的 M 和 V 意思一样。C 即 Controller 指的是页面业务逻辑，使用 MVC 的目的就是将 M 和 V 的代码分离。MVC 是单向通信。也就是 View 跟 Model，必须通过 Controller 来承上启下

**2、使用场景**

主要就是 MVC 中 Controller 演变成 MVVM 中的 viewModel，MVVM 主要解决了 MVC中大量的 DOM 操作使页面渲染性能降低，加载速度变慢，影响用户体验，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，需要大量操作 DOM 元素时，采用 MVVM 的开发方式，会更加便捷，让开发者更多的精力放在数据的变化上，解放繁琐的操作 DOM 元素

**3、两者之间的区别**

MVC 和 MVVM 其实区别并不大，都是一种设计思想， MVC 和 MVVM 的区别并不是VM 完全取代了 C，只是在 MVC 的基础上增加了一层 VM，只不过是弱化了 C 的概念，ViewModel 存在目的在于抽离 Controller 中展示的业务逻辑，而不是替代 Controller，其它视图操作业务等还是应该放在 Controller 中实现，也就是说 MVVM 实现的是业务逻辑组件的重用, 使开发更高效，结构更清晰，增加代码的复用性

4、Vue 数据双向绑定的原理是什么?（必会）

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。

1、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

2、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

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

3.1）在自身实例化时往属性订阅器(dep)里面添加自己

3.2）自身必须有一个 update()方法

3.3）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。

4、MVVM 作为数据绑定的入口，整合 Observer、Compile 和 Watcher 三者，通 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

5、Object.defineProperty 和 Proxy 的区别（必会）

**Object.defineProperty 和 Proxy 的区别如下:**

1、Proxy 可以直接监听对象而非属性；

2、Proxy 可以直接监听数组的变化；

3、Proxy 有多达 13 种拦截方法,不限于 apply、ownKeys、deleteProperty、has 等等 是 Object.defineProperty 不具备的

4、Proxy 返回的是一个新对象,我们可以只操作新的对象达到目的,而 Object.defineProperty 只能遍历对象属性直接修改

5、Proxy 作为新标准将受到浏览器厂商重点持续的性能优化，也就是传说中的新标准 的性能红利

6、Object.defineProperty 兼容性好，支持 IE9，而 Proxy 的存在浏览器兼容性问题, 而且无法用 polyfill 磨平，因此 Vue 的作者才声明需要等到下个大版本( 3.0 )才能用 Proxy 重写

6、Vue 生命周期总共分为几个阶段？（必会）

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。

1、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

2、created

在实例创建完成后被立即调用。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。然而，挂载阶段还没开始，$el 属性目前不可见

3、beforeMount

在挂载开始之前被调用：相关的 render 函数首次被调用

4、mounted

el 被新创建的 vm.$el 替换，并挂载到实例上去之后调用该钩子，如果 root 实例挂载了一个文档内元素，当 mounted 被调用时 vm.$el 也在文档内

5、beforeUpdate

数据更新时调用，发生在虚拟 DOM 打补丁之前。这里适合在更新之前访问现有的 DOM，比如手动移除已添加的事件监听器，该钩子在服务器端渲染期间不被调用，因为只有初次渲染会在服务端进行

6、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

7、activated

keep-alive 组件激活时调用。该钩子在服务器端渲染期间不被调用

8、deactivated

keep-alive 组件停用时调用。该钩子在服务器端渲染期间不被调用

9、beforeDestroy

实例销毁之前调用。在这一步，实例仍然完全可用。该钩子在服务器端渲染期间不被调用

10、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

11、errorCaptured（2.5.0+ 新增）

当捕获一个来自子孙组件的错误时被调用。此钩子会收到三个参数：错误对象、发生错误的组件实例以及一个包含错误来源信息的字符串，此钩子可以返回 false 以阻止该错误继续向上传播

7、第一次加载页面会触发哪几个钩子函数？（必会）

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

子函数

8、请说下封装 Vue 组件的过程？（必会）

首先，组件可以提升整个项目的开发效率。能够把页面抽象成多个相对独立的模块，解决

了我们传统项目开发：效率低、难维护、复用性等问题

1、分析需求：确定业务需求，把页面中可以服用的结构，样式以及功能，单独抽离成一个文件，实现复用

2、具体步骤：使用 Vue.extend 方法创建一个组件，然后使用 Vue.component 方法注册组件，子组件需要数据，可以在 props 中接受定义，而子组件修改好数据后，想把数据传递给父组件，可以采用$emit 方法

9、Vue 组件如何进行传值的? （必会）

1、父组件向子组件传递数据

父组件内设置要传的数据，在父组件中引用的子组件上绑定一个自定义属性并把数据 绑定在自定义属性上，在子组件添加参数 props 接收即可

2、子组件向父组件传递数据

子组件通过 vue 实例方法$emit 进行触发并且可以携带参数，父组件监听使用@（v-on）进行监听，然后进行方法处理

3、非父子组件之间传递数据

3.1 引入第三方 new vue 定义为 eventBus

3.2）在组件中 created 中订阅方法 eventBus.$on("自定义事件名",methods 中的方法 名)

3.3) 在另一个兄弟组件中的 methods 中写函数，在函数中发布 eventBus 订阅的方法 eventBus.$emit("自定义事件名”)

3.4) 在组件的 template 中绑定事件(比如 click)

10、组件中写 name 选项有什么作用？（必会）

1、项目使用 keep-alive 时，可搭配组件 name 进行缓存过滤

2、DOM 做递归组件时需要调用自身 name

3、vue-devtools 调试工具里显示的组见名称是由 vue 中组件 name 决定的

11、Vue 组件 data 为什么必须是函数（必会）

1、个组件都是 Vue 的实例

2、组件共享 data 属性，当 data 的值是同一个引用类型的值时，改变其中一个会影响其他

3、组件中的 data 写成一个函数，数据以函数返回值形式定义，这样每复用一次组件，就

会返回一份新的 data，类似于给每个组件实例创建一个私有的数据空间，让各个组件实例维护

各自的数据。而单纯的写成对象形式，就使得所有组件实例共用了一份 data，就会造成一个变

了全都会变的结果

12、讲一下组件的命名规范（必会）

给组件命名有两种方式，一种是使用链式命名 my-component，一种是使用大驼峰命名 MyComponent 在字符串模板中<my-component></my-component> 和

<MyComponent></MyComponent>都可以使用，在非字符串模板中最好使用

<MyComponent></MyComponent>，因为要遵循 W3C 规范中的自定义组件名 (字母全小写且

必须包含一个连字符)，避免和当前以及未来的 HTML 元素相冲突

13、怎么在组件中监听路由参数的变化？（必会）

有两种方法可以监听路由参数的变化，但是只能用在包含<router-view />的组件内。

第一种

watch: {

'$route'(to, from) {

// 在此处监听

},

},

第二种

beforeRouteUpdate (to, from, next) {

//这里监听

}

14、怎么捕获 Vue 组件的错误信息？（必会）

1、errorCaptured 是组件内部钩子，当捕获一个来自子孙组件的错误时被调用，接收error、vm、info 三个参数，return false 后可以阻止错误继续向上抛出

2、errorHandler 为全局钩子，使用 Vue.config.errorHandler 配置，接收参数与 errorCaptured 一致，2.6 后可捕捉 v-on 与 promise 链的错误，可用于统一错误处理与错误兜底

15、Vue 组件里的定时器要怎么销毁？（必会）

如果页面上有很多定时器，可以在 data 选项中创建一个对象 timer，给每个定时器取个名

字一一映射在对象 timer 中， 在 beforeDestroy 构造函数中

for(let k in this.timer){clearInterval(k)}；

如果页面只有单个定时器，可以这么做

const timer = setInterval(() =>{}, 500);

this.$once('hook:beforeDestroy', () => {

clearInterval(timer);

})

16、Vue-cli 用自定义的组件？有遇到过哪些问题吗？（必

会）

1、在 components 目录新建你的组件文件（indexPage.vue），script 一定要 export default

{}

2、在需要用的页面（组件）中导入：import indexPage from '@/components/indexPage.vue'

3、注入到 vue 的子组件的 components 属性上面,components:{indexPage}

4、在 template 视图 view 中使用，例如有 indexPage 命名，使用的时候则 index-page

17、Vue 中 solt 的使用方式，以及 solt 作用域插槽的用法

（必会）

**使用方式**

当组件当做标签进行使用的时候，用 slot 可以用来接受组件标签包裹的内容，当给 solt 标签添加 name 属性的 时候，可以调换响应的位置

**插槽作用域**

作用域插槽其实就是带数据的插槽，父组件接收来自子组件的 slot 标签上通过 v-bind绑定进而传递过来的数 据，父组件通过 scope 来进行接受子组件传递过来的数据

18、Vue 该如何实现组件缓存?（必会）

在面向组件化开发中，我们会把整个项目拆分为很多业务组件，然后按照合理的方式组

织起来，那么自然会存在组件之前切换的问题，vue 中有个动态组件的概念，它能够帮助开发

者更好的实现组件之间的切换，但是在面对需求频繁的变化，去要切换组件时，动态组件在切

换的过程中，组件的实例都是重新创建的，而我们需要保留组件的状态，为了解决这个问题，

需要使用到 vue 中内置组件<keep-alive>

<keep-alive></keep-alive> 包裹动态组件时，会缓存不活动的组件实例,主要用于保留组

件状态或避免重新渲染，

简单的说: 比如有一个列表和一个详情，那么用户就会经常执行打开详情=>返回列表=>打

开详情…这样的话列表和详情都是一个频率很高的页面，那么就可以对列表组件使用<keep

alive></keep-alive>进行缓存，这样用户每次返回列表的时候，都能从缓存中快速渲染，而不

是重新渲染

19、跟 keep-alive 有关的生命周期是哪些？（必会）

1、前言：在开发 Vue 项目的时候，大部分组件是没必要多次渲染的，所以 Vue 提供了一

个内置组件 keep-alive 来缓存组件内部状态，避免重新渲染，在开发 Vue 项目的时候，大部分组件是没必要多次渲染的，所以 Vue 提供了一个内置组件 keep-alive 来缓存组件内部状态，避免重新渲染

2、生命周期函数：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。

2.1）activated 钩子：在在组件第一次渲染时会被调用，之后在每次缓存组件被激活 时调用。

2.2）Activated 钩子调用时机：第一次进入缓存路由/组件，在 mounted 后面， beforeRouteEnter 守卫传给 next 的回调函数之前调用，并且给因为组件被缓存了，再次进入缓存路由、组件时，不会触发这些钩子函数，beforeCreate created beforeMount mounted 都不会触发

2.3）deactivated 钩子：组件被停用（离开路由）时调用：deactivated 钩子调用时 机：使用 keep-alive 就不会调用 beforeDestroy(组件销毁前钩子)和 destroyed(组件销毁)，因为组件没被销毁，被缓存起来了，这个钩子可以看作 beforeDestroy 的替代，如果你缓存了组件，要在组件销毁的的时候做一些事情，可以放在这个钩子里，组件内的离开当前路由钩子beforeRouteLeave => 路由前置守卫 beforeEach =>全局后置钩子 afterEach => deactivated离开缓存组件 => activated 进入缓存组件(如果你进入的也是缓存路由)

20、为什么避免 v-if 和 v-for 用在一起（必会）

当 Vue 处理指令时，v-for 比 v-if 具有更高的优先级，通过 v-if 移动到容器元素，不会

再重复遍历列表中的每个值。取而代之的是，我们只检查它一次，且不会在 v-if 为否的时候运

算

21、watch、methods 和 computed 的区别?（必会）

1、基本说明

1.1)computed：

计算属性将被混入到 Vue 实例中,所有 getter 和 setter 的 this 上下文自动地绑 定为 Vue 实例

1.2)methods：

methods 将被混入到 Vue 实例中。可以直接通过 VM 实例访问这些方法，或者 在指令表达式中使用。方法中的 this 自动绑定为 Vue 实例。

1.3)watch：

观察和响应 Vue 实例上的数据变动，一个对象，键是需要观察的表达式，值是对 应回调函数,值也可以是方法名，或者包含选项的对象，Vue 实例将会在实例化时调,$watch()，遍历 watch 对象的每一个属性

2、三者的加载顺序

2.1)computed 是在 HTML DOM 加载后马上执行的，如赋值；（属性将被混入到 Vue 实 例）

2.2)methods 则必须要有一定的触发条件才能执行，如点击事件，watch 呢？它用于观 察 Vue 实例上的数据变动，

2.3）默认加载的时候先 computed 再 watch，不执行 methods；

2.4）触发某一事件后先 computed 再 methods 再到 watch，computed 属性 vs method 方，computed 计算属性是基于它们的依赖进行缓存的

3、总结

计算属性 computed 只有在它的相关依赖发生改变时才会重新求值，当有一个性能开

销比较大的的计算属性 A ，它需要遍历一个极大的数组和做大量的计算，然后我们可能有其

他的计算属性依赖于 A ，这时候，我们就需要缓存，每次确实需要重新加载，不需要缓存时

用 methods

22、怎么在 watch 监听开始之后立即被调用？（必会）

在选项参数中指定 immediate: true 将立即以表达式的当前值触发回调

23、computed 中的属性名和 data 中的属性名可以相同 吗？（必会）

不能同名，因为不管是 computed 属性名还是 data 数据名还是 props 数据名都会被挂载在 vm 实例上，因此这三个都不能同名

if (key in vm.$data) {

warn(`The computed property "${key}" is already defined in data.`, vm)

} else if (vm.$options.props && key in vm.$options.props) {

warn(`The computed property "${key}" is already defined as a prop.`, vm)

24、什么是 Vue 的计算属性（必会）

在模板中放入太多的逻辑会让模板过重且难以维护，在需要对数据进行复杂处理，且可

能多次使用的情况下，尽量采取计算属性的方式，好处：使得数据处理结构清晰；

依赖于数据，数据更新，处理结果自动更新；

1、计算属性内部 this 指向 vm 实例

2、在 template 调用时，直接写计算属性名即可

3、常用的是 getter 方法，获取数据，也可以使用 set 方法改变数据

4、相较于 methods，不管依赖的数据变不变，methods 都会重新计算，但是依赖数据不变的时候 computed 从缓存中获取，不会重新计算

25、Vue 中 key 值的作用是什么？（必会）

当 Vue.js 用 v-for 正在更新已渲染过的元素列表时，它默认用“就地复用”策略。如果数据项的顺序被改变，Vue 将不会移动 DOM 元素来匹配数据项的顺序， 而是简单复用此处每个元素，并且确保它在特定索引下显示已被渲染过的每个元素，key 的作用主要是为了高效的更新虚拟 DOM

26、Vue-loader 是什么？使用它的用途有哪些？（必会）

vue-loader 会解析文件，提取出每个语言块，如果有必要会通过其他 loader 处理，最后将 他们组装成一个 commonjs 模块；module.exports 出一个 vue.js 组件对象

1、< temlate>语言块

1,1）默认语言：html

1,2）每个.vue 文件最多包含一个< template>块

1,3）内容将被提取为字符串，将编译用作 VUE 组件的 template 选项

2、< script>

2,1）默认语言：

JS（在监测到 babel-loader 或者 buble-loader 配置时，自动支持ES2015）

2,2）每个.vue 文件最多包含一个< script>块

2,3）该脚本在类 CommonJS 环境中执行（就像通过 webpack 打包的正常 JS 模块）。所以你可以 require()其他依赖。在 ES2015 支持下，也可以使用 import 跟 export 语法

2,4）脚本必须导出 Vue.js 组件对象，也可以导出由 VUE.extend()创建的扩展对象；但是普通对象是更好的选择

3、< style>

默认语言：css

3,1）一个.vue 文件可以包含多个< style>标签

3,2）这个标签可以有 scoped 或者 module 属性来帮助你讲样式封装到当前组件；具有不同封装模式的多个< style>标签可以在同一个组件中混合使用

3,3）默认情况下，可以使用 style-loader 提取内容，并且通过< style>标签动态假如文档的< head>中，也可以配置 webpack 将所有的 styles 提取到单个 CSS 文件中

4、自定义块

可以在.vue 文件中添加额外的自定义块来实现项目的特殊需求；例如< docs>块；vue-loader 将会使用标签名来查找对应的 webpack loaders 来应用到对应的模块上；webpack 需要在 vue-loader 的选项 loaders 中指定vue-loader 支持使用非默认语言，比如 CSS 预处理器，预编译的 HTML 模板语言，通过设置语言块的 lang 属性：

<style lang='sass'>

/\*sass\*/

</style>

27、Vue 中怎么自定义过滤器（必会）

Vue.js 允许自定义过滤器，可被用于一些常见的文本格式化。过滤器可以用在两个地方：

双花括号插值和 v-bind 表达式。过滤器应该被添加在 JavaScript 表达式的尾部，由“管道”符

号指示可以用全局方法 Vue.filter() 注册一个自定义过滤器，它接收两个参数：过滤器 ID 和过滤器函数。过滤器函数以值为参数，返回转换后的值

Vue .filter( 'reverse' , function (value) { return value.split( '' ).reverse().join( '' ) })

<span v-text = "message | reverse"></span>

过滤器也同样接受全局注册和局部注册

28、你是怎么认识 Vuex 的?（必会）

vuex 可以理解为一种开发模式或框架。比如 PHP 有 thinkphp，java 有 spring 等，通过状态(数据源)集中管理驱动组件的变化(好比 spring 的 IOC 容器对 bean 进行集中管理)

1. 应用级的状态集中放在 store 中
2. 改变状态的方式是提交 mutations，这是个同步的事物

3、异步逻辑应该封装在 action 中

29、Vuex 的 5 个核心属性是什么?（必会）

分别是 State、 Getter、Mutation 、Action、 Module

1、state

state 为单一状态树，在 state 中需要定义我们所需要管理的数组、对象、字符串等等， 只有在这里定义了，在 vue.js 的组件中才能获取你定义的这个对象的状态

2、getter

getter 有点类似 vue.js 的计算属性，当我们需要从 store 的 state 中派生出一些状态， 那么我们就需要使用 getter，getter 会接收 state 作为第一个参数，而且 getter 的返回值会根据它的依赖被缓存起来，只有 getter 中的依赖值（state 中的某个需要派生状态的值）发生改变的时候才会被重新计算

3、mutation

更改 store 中 state 状态的唯一方法就是提交 mutation，就很类似事件。每个 mutation都有一个字符串类型的事件类型和一个回调函数，我们需要改变 state 的值就要在回调函数中改变。我们要执行这个回调函数，那么我们需要执行一个相应的调用方法store.commit

4、action

action 可以提交 mutation，在 action 中可以执行 store.commit，而且 action 中可以有任何的异步操作。在页面中如果我们要嗲用这个 action，则需要执行 store.dispatch

5、module

module 其实只是解决了当 state 中很复杂臃肿的时候，module 可以将 store 分割成模块，每个模块中拥有自己的 state、mutation、action 和 getter

30、Vuex 的出现解决了什么问题?（必会）

主要解决了以下两个问题

1、多个组件依赖于同一状态时，对于多层嵌套的组件的传参将会非常繁琐，并且对于兄弟组件间的状态传递无能为力

2、来自不同组件的行为需要变更同一状态。以往采用父子组件直接引用或者通过事件来变更和同步状态的多份拷贝。以上的这些模式非常脆弱，通常会导致无法维护的代码

31、简述 Vuex 的数据传递流程（必会）

当组件进行数据修改的时候我们需要调用 dispatch 来触发 actions 里面的方法。actions 里面的每个方法中都会 有一个

1、commit 方法，当方法执行的时候会通过 commit 来触 mutations 里面的方法进行数据 的修改

2、mutations 里面的每个函数都会有一个 state 参数，这样就可以在 mutations 里面进行 state 的数据修改 ，当数据修改完毕后，会传导给页面，页面的数据也会发生改变

32、Vuex 的 Mutation 和 Action 之间的区别是什么？（必

会）

1、流程顺序

“相应视图—>修改 State”拆分成两部分，视图触发 Action，Action 再触发 Mutation

2、角色定位

基于流程顺序，二者扮演不同的角色

1）Mutation：专注于修改 State，理论上是修改 State 的唯一途径

2）Action：业务代码、异步请求

3、限制

1）角色不同，二者有不同的限制

2）Mutation：必须同步执行

3）Action：可以异步，但不能直接操作 State

33、Vue-router 是干什么的，原理是什么？（必会）

Vue-router 是 Vue.js 官方的路由插件，它和 vue.js 是深度集成的，适合用于构建单页面应用，vue 的单页面应用是基于路由和组件的，路由用于设定访问路径，并将路径和组件映射起来，传统的页面应用，是用一些超链接来实现页面切换和跳转的，在 vue-router 单页面应用中，则是路径之间的切换，也就是组件的切换，路由模块的本质 就是建立起 url 和页面之间的映射关系“更新视图但不重新请求页面”是前端路由原理的核心之一，目前在浏览器环境中这一功

能的实现主要有两种方式：

1、利用 URL 中的 hash（“#”）

2、利用 History interface 在 HTML5 中新增的方法

34、路由之间是怎么跳转的？有哪些方式？（必会）

1、<router-link to="需要跳转到页面的路径">

2、this.$router.push()跳转到指定的 url，并在 history 中添加记录，点击回退返回到上一个 页面

3、this.$router.replace()跳转到指定的 url，但是 history 中不会添加记录，点击回退到上上

个页面

4、this.$touter.go(n)向前或者后跳转 n 个页面，n 可以是正数也可以是负数

35、Vue-router 怎么配置路由（必会）

在 vue 中配置路由分为 5 个步骤，分别是：

1、安装

npm install --save vue-router

2、引用

import VueRouter from 'vue-router'

3、配置路由文件

var router = new VueRouter({

routes:[

{

path:"/hello",

component:HelloWorld

},

{

path:"/wen",

component:HelloWen

}

]

new Vue({

el: '#app',

components: { App },

router,

template: '<App/>'

})

4、视图加载的位置

默认 App.vue 文件中加<router-view></router-view>

5、跳转导航

<router-link to="/hello">helloword</router-link>（渲染出来的是 a 标签）

36、Vue-router 有哪几种路由守卫?（必会）

1、路由守卫为

2、全局守卫：beforeEach

3、后置守卫：afterEach

4、全局解析守卫：beforeResolve

5、路由独享守卫：beforeEnter

37、Vue-router 的钩子函数都有哪些?（必会）

关于 vue-router 中的钩子函数主要分为 3 类

1、全局钩子函数要包含 beforeEach

1,1）beforeEach 函数有三个参数,分别是

1,2）to:router 即将进入的路由对象

1,3）from:当前导航即将离开的路由

1,4）next:function,进行管道中的一个钩子，如果执行完了,则导航的状态就是 confirmed （确认的）否则为 false,终止导航

2、单独路由独享组件

2,1）beforeEnter

3、组件内钩子

3,1）beforeRouterEnter

3,2）beforeRouterUpdate

3,3）beforeRouterLeave

38、路由传值的方式有哪几种(必会)

Vue-router 传参可以分为两大类，分别是编程式的导航 router.push 和声明式的导航

1、router.push

1.1）字符串：直接传递路由地址，但是不能传递参数this.$router.push("home")

对象：

1.2）命名路由 这种方式传递参数，目标页面刷新会报错 this.$router.push({name:"news",params:{userId:123})

1.3）查询参数 和 name 配对的式 params，和 path 配对的是 query this.$router.push({path:"/news',query:{uersId:123})

1.4）接收参数 this.$route.query

2、声明式导航

2.1）字符串 <router-link to:"news"></router-link>

2.2）命名路由 <router-link :to:"{name:'news',params:{userid:1111}}"></route-link>

2.3）查询参数 <router-link :to="{path:'/news',query:{userId:1111}}"></router-link>

39、怎么定义 Vue-router 的动态路由?怎么获取传过来的动

态参数?

我们经常需要把某种模式匹配到的所有路由，全都映射到同个组件，例如，我们有一个

User 组件，对于所有 ID 各不相同的用户，都要使用这个组件来渲染，那么，我们可以在

vue-router 的路由路径中使用“动态路径参数”(dynamic segment) 来达到这个效果

1、动态路径参数，使用“冒号”开头，一个路径参数，使用冒号标记，当匹配到一个路由时，参数会被设置到 this.$router.params 中，并且可以在每个组件中使用

2、现在我们知道了可以通过动态路由传参，在路由中设置了，多段路径参数后，对应的

值分别都会设置到$router.query 和$router.params 中

40、query和 params 之间的区别是什么？（必会）

1、query 要用 path 来引入，params 要用 name 来引入

2、接收参数时，分别是 this.$route.query.name 和 this.$route.params.name（注意：是 $route 而不是$router

3、query 更加类似于我们 ajax 中 get 传参，params 则类似于 post，前者在浏览器的地址栏中显示，params 不显示

4、params 传值一刷新就没了，query 传值刷新还存在

41、$route 和$router 的区别是什么？（必会）

$route 是“路由信息对象”，包括 path，params，hash，query，fullPath，matched，name 等路由信息参数

$router 为 VueRouter 的实例，相当于一个全局的路由器对象，里面含有很多属性和子对 象，例如 history 对象，经常用的跳转链接就可以用 this.router.push 会往 history 栈中添加一个新的记录。返回上一个 history 也是使用$router.go 方法

42、active-class 属于哪个组件中的属性？该如何使用？

首先 active-class 是 vue-router 模块中 router-link 组件中的属性，主要作用是用来实现选中样式的切换，在 vue-router 中要使用 active-class 有两种方式：

1、 在 router-link 中写入 active-class active-class 选择样式时根据路由中的路径（to=“

/home”）去匹配，然后显示<router-link to="/home" class="menu-home" active-class="active">首页</router-link>

2、直接在路由 js 文件中配置 linkActiveClass

export default new Router({

linkActiveClass: 'active',

})

<div class="menu-btn">

<router-link to="/" class="menu-home" active-class="active">

首页

</router-link>

</div>

<div class="menu-btn">

<router-link to="/my" class="menu-my" active-class="active">

我的

</router-link>

</div>

3、引起的问题

因为 to="/" 引起的，active-class 选择样式时根据路由中的路径去匹配，然后显示， 例如在 my 页面中，路由为 localhost:8081/#/my，那么 to="/”和 to="/my"都可以匹配到，所有都会激活选中样式

4、解决方法

4,1）在 router-link 中写入 exact

<router-link to="/" class="menu-home" active-class="active" exact>首页 </router-link>

4,2）在路由中加入重定向

<router-link to="/" class="menu-home" active-class="active" exact>首页 </router-link>

{

path: '/',

redirect: '/home'

}

43、Vue 的路由实现模式：hash 模式和 history 模式（必

会）

1、hash 模式：在浏览器中符号“#”，#以及#后面的字符称之为 hash，用 window.location.hash 读取。特点：hash 虽然在 URL 中，但不被包括在 HTTP 请求中；用来指导浏览器动作，对服务端安全无用，hash 不会重加载页面

2、history 模式：history 采用 HTML5 的新特性，且提供了两个新方法：

2.1）pushState()

2.2）replaceState()可以对浏览器历史记录栈进行修改，以及 popState 事件的监听到状态变更

44、请说出路由配置项常用的属性及作用（必会）

路由配置参数：

1、path : 跳转路径

2、component : 路径相对于的组件

3、name:命名路由

4、children:子路由的配置参数(路由嵌套)

5、props:路由解耦

6、redirect : 重定向路由

45、编程式导航使用的方法以及常用的方法（必会）

1、路由跳转：this.$router.push()

2、路由替换: this.$router.replace()

3、后退：this.$router.back()

4、前进 ：this.$router.forward()

46、Vue 怎么实现跨域（必会）

1、什么是跨域

跨域指浏览器不允许当前页面的所在的源去请求另一个源的数据。源指协议，端口，域名。只要这个 3 个中有一个不同就是跨域

2、使用 vue-cli 脚手架搭建项目时 proxyTable 解决跨域问题

打开 config/index.js,在 proxyTable 中添写如下代码：

proxyTable: {

'/api': { //使用"/api"来代替"http://f.apiplus.c"

target: 'http://f.apiplus.cn', //源地址

changeOrigin: true, //改变源

pathRewrite: {

'^/api': 'http://f.apiplus.cn' //路径重写

}

3、使用 CORS（跨域资源共享）

3.1）前端设置，vue 设置 axios 允许跨域携带 cookie（默认是不带 cookie） axios.defaults.withCredentials = true;

3,2）后端设置：

3.2.1）跨域请求后的响应头中需要设置

3.2.2）Access-Control-Allow-Origin 为发起请求的主机地址

3.2.3）Access-Control-Allow-Credentials，当它被设置为 true 时，允许跨域带 cookie，但此时 Access-Control- Allow-Origin 不能为通配符\*

3.2.4）Access-Control-Allow-Headers，设置跨域请求允许的请求头

3.2.5）Access-Control-Allow-Methods，设置跨域请求允许的请求方式

47、你对 Vue.js 的 template 编译的理解？（必会）

简而言之，就是先转化成 AST 树，再得到的 render 函数返回 Vnode（Vue 的虚拟 DOM 节点）

1、首先通过 compile 编译器把 template 编译成 AST 语法树（abstract syntax tree 即 源代码的抽象语法结构的树状表现形式），compile 是 createCompiler 的返回值，createCompiler 是用以创建编译器的，另外 compile 还负责合并 option

2、然后 AST 会经过 generate（将 AST 语法树转化成 render funtion 字符串的过程）得到render 函数，render 的返回值是 Vnode，Vnode 是 Vue 的虚拟 DOM 节点，里面有（标签名、子节点、文本等等）

48、Vue2.0 兼容 IE 哪个版本以上吗？（必会）

不支持 ie8 及以下，部分兼容 ie9 ，完全兼容 10 以上，因为 vue 的响应式原理是基于 es5的 Object.defineProperty(),而这个方法不支持 ie8 及以下

49、说一下你在 Vue 中踩过的坑（必会）

1、第一个是给对象添加属性的时候，直接通过给 data 里面的对象添加属性然后赋值，新

添加的属性不是响应式的

1.1）解决办法：通过 Vue.set(对象，属性，值)这种方式就可以达到，对象新添加的属 性是响应式的

2、在 created 操作 dom 的时候，是报错的，获取不到 dom，这个时候实例 vue 实例没有挂载

2.2）解决办法：通过：Vue.nextTick(回调函数进行获取)

50、Vue 项目优化的解决方案都有哪些？（必会）

1、 使用 mini-css-extract-plugin 插件抽离 css

2、 配置 optimization 把公共的 js 代码抽离出来

3、 通过 webpack 处理文件压缩

4、 不打包框架、库文件，通过 cdn 的方式引入

5、 小图片使用 base64

6、 配置项目文件懒加载

7、 UI 库配置按需加载

8、 开启 Gzip 压缩

51、使用 Vue 的时候一下加载造成页面卡顿，该如何解决？ （必会）

vue-router 解决首次加载缓慢的问题。懒加载简单来说就是按需加载

1、像 vue 这种单页面应用，如果没有应用懒加载，运用 webpack 打包后的文件将会异常的大，造成进入首页时， 需要加载的内容过多，时间过长，会出现长时间的白屏，即使做了 loading 也是不利于用户体验，

2、而运用懒加载 则可以将页面进行划分，需要的时候加载页面，可以有效的分担首页所承担的加载压力，减少首页加载用时。

3、用法：在配置路由时使用：component:resolve=>require([“@components/路由的路径”]，resolve)。 就是用了懒加载后打完包直接运行那个 index.html 会报错，报文件引用错误其实是打包时候路径配置有点问 题，找到 build 下面的webpack.prod.conf.js 添加 publicPath："./",

52、请说出 Vue.cli 项目中 src 目录每个文件夹和文件的用

法？（必会）

1、assets 文件夹是放静态资源

2、components 是放组件

3、router 是定义路由相关的配置

4、view 视图

5、app.vue 是一个应用主组件

6、main.js 是入口文件

53、说说你对 SPA 单页面的理解，它的优缺点分别是什么？（必会）

单页 Web 应用 (single-page application 简称为 SPA) 是一种特殊的 Web 应用，它将所有的活动局限于一个 Web 页面中，仅在该 Web 页面初始化时加载相应的 HTML、JavaScript

和 CSS，一旦页面加载完成了，SPA 不会因为用户的操作而进行页面的重新加载或跳转，取而

代之的是利用 JavaScript 动态的变换 HTML 的内容，从而实现 UI 与用户的交互，由于避免了

页面的重新加载，SPA 可以提供较为流畅的用户体验，得益于 ajax，我们可以实现无跳转刷新，又多亏了浏览器的 histroy 机制，我们用 hash 的变化从而可以实现推动界面变化，从而模拟元素客户端的单页面切换效果：

SPA 被人追捧是有道理的，但是它也有不足之处，当然任何东西都有两面性，以下是卤煮

总结的一些目前 SPA 的优缺点：

1、 优点：

1,1）无刷新界面，给用户体验原生的应用感觉

1,2）节省原生（android 和 ios）app 开发成本

1,3）提高发布效率，无需每次安装更新包

1,4）容易借助其他知名平台更有利于营销和推

1,5）符合 web2.0 的趋势

2、 缺点：

1） 效果和性能确实和原生的有较大差距

2） 各个浏览器的版本兼容性不一样

3） 业务随着代码量增加而增加，不利于首屏优化

4） 某些平台对 hash 有偏见，有些甚至不支持 pushstate

5） 不利于搜索引擎抓取

54、怎样理解 Vue 的单向数据流？（必会）

1、数据从父级组件传递给子组件，只能单向绑定

2、子组件内部不能直接修改从父级传递过来的数据

3、所有的 prop 都使得其父子 prop 之间形成了一个单向下行绑定：父级 prop 的更新会向下流动到子组件中，但是反过来则不行，这样会防止从子组件意外改变父级组件的状态，从而导致你的应用的数据流向难以理解。

4、每次父级组件发生更新时，子组件中所有的 prop 都将会刷新为最新的值，这意味着你不应该在一个子组件内部改变 prop。如果你这样做了，Vue 会在浏览器的控制台中发出警告

5、子组件想修改时，只能通过 $emit 派发一个自定义事件，父组件接收到后，由父组件修改

55、VNone 是什么？什么是虚拟 DOM？（高薪常问）

**1、Vnode 是什么**

Vnode 是 JavaScript 对象，Vnode 表示 Virtual DOM，用 JavaScript 对象来描述真实的DOM 把 DOM 标签，属性，内容都变成对象的属性。就像使用 JavaScript 对象对一种动物进行说明一样{name: 'Hello Kitty', age: 1, children: null}。

1）Vnode 的作用

通过 render 将 template 模版描述成 Vnode，然后进行一系列操作之后形成真实的 DOM 进行挂载。

2) Vnode 的优点

2-1) 兼容性强，不受执行环境的影响。Vnode 因为是 JS 对象，不管 node 还是浏览器，都可以统一操作，从而获得了服务端渲染、原生渲染、手写渲染函数等能力。

2-2) 减少操作 DOM，任何页面的变化，都只使用 Vnode 进行操作对比，只需要在最后一步挂载更新 DOM，不需要频繁操作 DOM，从而提高页面性能。

**3、 什么是虚拟 DOM?**

1、文档对象模型或 DOM 定义了一个接口，该接口允许 JavaScript 之类的语言访问和操作 HTML 文档。元素由树中的节点表示，并且接口允许我们操纵它们。但是此接口需要付出代价，大量非常频繁的 DOM 操作会使页面速度变的非常缓慢

2、Vue 通过在内存中实现文档结构的虚拟表示来解决此问题，其中虚拟节点（Vnode）表示 DOM 树中的节点。当需要操纵时，可以在虚拟 DOM 的 内存中执行计算和操作，而不是在真实 DOM 上进行操纵。这自然会更快，并且允许虚拟 DOM 算法计算出最优化的方式来更新实际 DOM 结构,一旦计算出，就将其应用于实际的 DOM 树，这就提高了性能，这就是为什么基于虚拟 DOM 的框架（例如 Vue 和 react）如此突出的原因。

56、Vue 中如何实现一个虚拟 DOM？说说你的思路（高薪 常问）

首先要构建一个 Vnode 的类，DOM 元素上的所有属性在 Vnode 类实例化出来的对象上都存在对应的属性。例如 tag 表示一个元素节点的名称，text 表示一个文本节点的文本，chlidren

表示子节点等。将 Vnode 类实例化出来的对象进行分类，例如注释节点、文本节点、元素节

点、组件节点、函数式节点、克隆节点。然后通过编译将模板转成渲染函数 render，执行渲染函数 render，在其中创建不同类型的Vnode 类，最后整合就可以得到一个虚拟 DOM（vnode），最后通过 patch 将 vnode 和oldVnode 进行比较后，生成真实 DOM

57、Vue 中操作 data 中数组的方法中哪些可以触发视图更 新，哪些不可以，不可以的话有什么解决办法？（高薪常问）

1、可以被改变的

push()、pop()、shift()、unshift()、splice()、sort()、reverse()这些方法会改变被操作的数 组；

2、不可以改变的

filter()、concat()、 slice()这些方法不会改变被操作的数组，并且返回一个新的数组，以上方法都可以触发视图更新

3、解决方案

1、利用索引直接设置一个数组项，例：this.array[index] = newValue，直接修改数组的长度，例：this.array.length = newLength

2、以上两种方法不可以触发视图更新

1）可以使用 this.$set(this.array,index,newValue) this.array.splice(index,1,newValue)

2）可以使用 this.array.splice(newLength)

58、Vue 中如何重置 data? （高薪常问）

要初始化 data 中的数据，可以使用 Object.assign()方法，实现重置 data 中的数据，以下就是对该方法的详细介绍，以及如何使用该方法，重置 data 中的数据

1、Object.assign()方法基本定义

1，1）Object.assign() 方法用于将所有可枚举属性的值从一个或多个源对象复制到目 标对象。它将返回目标对象。

1，2）用法：Object.assign(target, ...sources)，第一个参数是目标对象，第二个参数是源对象，就是将源对象属性复制到目标对象，返回目标对象

2、具体使用方式

使用 Object.assign()，vm.$data 可以获取当前状态下的 data，vm.$options.data(this)

可以获取到组件初始化状态下的 data，复制 Object.assign(this.$data, this.$options.data(this))

// 注意加 this，不然取不到 data() { a: this.methodA } 中的 this.methd

59、如何对 Vue 首屏加载实现优化? （高薪常问）

1、把不常改变的库放到 index.html 中，通过 cdn 引入

2、vue 路由的懒加载

3、不生成 map 文件

4、vue 组件尽量不要全局引入

5、使用更轻量级的工具库

6、开启 gzip 压缩

7、首页单独做服务端渲染

60、Vue 的 nextTick 的原理是什么? （高薪常问）

1、为什么需要 nextTick

Vue 是异步修改 DOM 的并且不鼓励开发者直接接触 DOM，但有时候业务需要必须对数据更改--刷新后的 DOM 做相应的处理，这时候就可以使用 Vue.nextTick(callback)这个api 了

2、理解原理前的准备

首先需要知道事件循环中宏任务和微任务这两个概念

2,1）常见的宏任务有：script, setTimeout, setInterval, setImmediate, I/O, UI rendering

2,2）常见的微任务有：process.nextTick(nodejs),Promise.then(), MutationObserver

3、理解 nextTick 的原理

正是 vue 通过异步队列控制 DOM 更新和 nextTick 回调函数先后执行的方式。如果大家看过这部分的源码，会发现其中做了很多 isNative()的判断，因为这里还存在兼容性优雅降级的问题