## vue中的组件通讯

1、props和$emit

父组件向子组件传递数据是通过prop传递的，子组件传递数据给父组件是通过$emit触发事件

2、$attrs和$listeners

3、中央事件总线 bus

上面两种方式处理的都是父子组件之间的数据传递，而如果两个组件不是父子关系呢？这种情况下可以使用中央事件总线的方式。新建一个Vue事件bus对象，然后通过bus.$emit触发事件，bus.$on监听触发的事件。

4、provide和inject

父组件中通过provider来提供变量，然后在子组件中通过inject来注入变量。不论子组件有多深，只要调用了inject那么就可以注入provider中的数据。而不是局限于只能从当前父组件的prop属性来获取数据，只要在父组件的生命周期内，子组件都可以调用。

6、$parent和$children

7、boradcast和dispatch

8、vuex处理组件之间的数据交互 如果业务逻辑复杂，很多组件之间需要同时处理一些公共的数据，这个时候才有上面这一些方法可能不利于项目的维护，vuex的做法就是将这一些公共的数据抽离出来，然后其他组件就可以对这个公共数据进行读写操作，这样达到了解耦的目的。

## vue父子组件传值？

父传子：子用props接父传过来的值，详细说：父组件注册子组件时候，给子组件绑定一个自定义属性，子用props接，props是单向绑定的，不限制数据类型

子传父：子组件向父组件传值是通过$emit触发事件做到。详细说：在父组件中注册子组件，给子组件绑定并监听一个自定义事件，在子组件中使用$emit触发刚才的自定义事件，$emit(事件名，参数)，父组件中的自定义事件对应的回调函数就能接收到子组件传来的数据，on去监听事件

在使用过程中想到某种情况，即在父组件中调用子组件$emit的方法时，还要加上父组件的参数，该如何处理？

比如上面父组件中，@showCityName=“updateCity”，updateCity中自带有子组件中传递的参数；但是如果尝试添加一个来自父组件的参数（比如v-for中的index），写成@showCityName=“updateCity(i)”，methods中的updateCity(data)里发现，data = i。开始怀疑是漏掉了参数，于是把updateCity函数改写成updateCity(index,data)，测试发现，index = i，data = undefined，而改成updateCity(…data)，结果data = {i}。

## vue路由跳转

router-link 带参数和不带参数

1.this.$router.push()

2.this.$router.replace()

3.this.$router.go(n)

this.$router.push

跳转到指定url路径，并想history栈中添加一个记录，点击后退会返回到上一个页面

this.$router.replace

跳转到指定url路径，但是history栈中不会有记录，点击返回会跳转到上上个页面 (就是直接替换了当前页面)

this.$router.go(n)

向前或者向后跳转n个页面，n可为正整数或负整数

### 二、$ router和$ route的区别

$router : 是路由操作对象，只写对象

$route : 路由信息对象，只读对象

$ router操作路由跳转

this.$router.push({ name:‘hello’, params:{ name:‘word’, age:‘11’ } })

$route读取 路由参数接收

var name = this.$route.params.name;

**vue路由守卫**

2）、路由独享守卫

写在路由配置里。[钩子函数](https://so.csdn.net/so/search?q=%E9%92%A9%E5%AD%90%E5%87%BD%E6%95%B0&spm=1001.2101.3001.7020" \t "https://note.youdao.com/newEditorV1/_blank)名：beforeEnter，只有前置守卫

3）、组件内部守卫

写在组件对象里。分别有前置守卫，后置守卫，路由改变守卫（当前组件被复用的情况,不是路径改变）三个钩子函数。

## vue页面级组建之间传值

1.使用vue-router通过跳转链接带参传值

2.使用本地存储localStorge

3.使用vuex数据管理传值

## 解释下vue的双向数据绑定

vue内部使用object.defineProperty方法给所有数据加上getter和setter方法，数据有变化时，通知订阅者watcher，watcher会触发它的update方法，对视图进行更新。也就是说数据发生变化时，视图跟着变化。

弊端：数组拿到index下标值修改后原来的不会拿到 所以出现了set方法

## 关于对象

Vue 无法检测 property 的添加或移除。由于 Vue 会在初始化实例时对 property 执行 getter/setter 转化，所以 property 必须在 data 对象上存在才能让 Vue 将它转换为响应式的。

对于已经创建的实例，Vue 不允许动态添加根级别的响应式 property。但是，可以使用

Vue.set(object, propertyName, value) 方法向嵌套对象添加响应式 property。

您还可以使用 vm.$set 实例方法，这也是全局 Vue.set 方法的别名

有时你可能需要为已有对象赋值多个新 property，比如使用 Object.assign() 或 \_.extend()。但是，这样添加到对象上的新 property 不会触发更新。在这种情况下，你应该用原对象与要混合进去的对象的 property 一起创建一个新的对象

## 关于数组

Vue 不能检测以下数组的变动：

1.当你利用索引直接设置一个数组项时，例如：vm.items[indexOfItem] = newValue

2.当你修改数组的长度时，例如：vm.items.length = newLength

[声明响应式 property](https://cn.vuejs.org/v2/guide/reactivity.html" \l "%E5%A3%B0%E6%98%8E%E5%93%8D%E5%BA%94%E5%BC%8F-property" \t "https://note.youdao.com/newEditorV1/_blank)

由于 Vue 不允许动态添加根级响应式 property，所以你必须在初始化实例前声明所有根级响应式 property，哪怕只是一个空值

如果你未在 data 选项中声明 message，Vue 将警告你渲染函数正在试图访问不存在的 property。

## computed和watch区别

computed：计算属性是基于它们的依赖进行缓存的，只有在它的相关依赖发生改变时才会重新求值。

watch监听对象需要深度监听，默认是浅监听

当页面中有某些数据依赖其他数据进行变动的时候，可以使用计算属性computed。

watch用于观察和监听页面上的vue实例，如果要在数据变化的同时进行异步操作或者是比较大的开销，那么watch为最佳选择。

## v-if和v-show什么区别

v-show通过css display控制显示和隐藏，v-if组件真正的渲染和销毁，而不是显示和隐藏，频繁切换状态使用v-show 否则v-if

v-if 常用于一次性改变，如根据权限决定是否显示

v-show 用于 tabs 切换

v-if 可与 templete块连用 ，v-show 不支持

## V-if v-for 优先级

v-for和v-if不应该一起使用，必要情况下应该替换成computed属性。原因：v-for比v-if优先，如果每一次都需要遍历整个数组，将会影响速度，尤其是当之需要渲染很小一部分的时候。

## Vue中v-for 中的 key 设置的值

快速查找到节点，减少渲染次数，提升渲染性能

## 有了解过key为什么不能使用index

当数据长度发生改变时，原来对应key的绑定关系会发生改变，所以被重新渲染，影响应能

## watch中能不能使用箭头函数

不能使用箭头函数定义watcher（回调）函数，因为箭头函数绑定了父级作用域的上下文，所以里面的 this 将不会按照期望指向 Vue 实例

watch是一个监听属性，监听data中name属性的变化，name两个参数 一个表示当前监听到的值，一个表示上一次监听到的值

## Vue响应式原理

1.描述监听data变化

监听对象变化：vue2.0核心api是Object.defineProperty，vue3.0是启用proxy实现响应式

一、Object.defineProperty的作用

Object.defineProperty方法直接在一个对象上定义一个新属性，或者修改一个已经存在的属性， 并返回这个对象。

二、语法

Object.defineProperty 需要三个参数（object , propName , descriptor）

1 object 对象 => 给谁加

2 propName 属性名 => 要加的属性的名字 【类型：String】

3 descriptor 属性描述 => 加的这个属性有什么样的特性【类型：Object】

监听数组变化：重写数组的

push.pop.shift.unshift.splice.sort.reverse方法

## 数组的哪些方法会引起视图更新 7个

push() pop() shift() unshift() splice() sort() reverse()

## vue的生命周期有啥

1、beforeCreate():组件实例刚刚被创建 (el 和 data 并未初始

化)

2、created():组件创建完成,属性已绑定,但 DOM 还未生成,$el

属性还不存在(完成 data 数据的初始化)

3、beforeMount():模板编译/挂载之前(完成了 el 和 data 初始

化)

4、Mounted():模板编译/挂载之后(完成挂载)

5、beforeUpdate():组件更新之前

6、updated()：组件更新之后

7、beforedestroy():组件销毁之前

8、destroyed():组件销毁之后

## vuex

vuex 是一个专门为 vue 构建的状态管理工具，主要是为了解决 多组间之间状态共享问题。强调的是集中式管理，（组件与组件之间的关系变成了组件与仓库之间的关系）

vuex的核心: **state（死得特）**(存放状态)，**mutations（缪忒神）**（同步的更改状态），**actions（啊可什）**(发送异步请求,拿到数据), **getters（该特死）**（根 据 之前的状态派发新的状态）, **modules（忙丢死）**（模块划分）

vuex的流程：state发布一条新的数据，在getters里面根据状态派发新的状态，actions发送异步请求获取数据，然后在mutations里面同步更新的数据

应用场景: 购物车的数据分享，登录注册

Vue Components：Vue组件。HTML页面上，负责接收用户操作等交互行为，执行dispatch方法触发对应action进行回应。

dispatch：操作行为触发方法，是唯一能执行action的方法。

actions：操作行为处理模块,由组件中的$store.dispatch('action 名称', data1)来触发。然后由commit()来触发mutation的调用 , 间接更新 state。负责处理Vue Components接收到的所有交互行为。包含同步/异步操作，支持多个同名方法，按照注册的顺序依次触发。向后台API请求的操作就在这个模块中进行，包括触发其他action以及提交mutation的操作。该模块提供了Promise的封装，以支持action的链式触发。

commit：状态改变提交操作方法。对mutation进行提交，是唯一能执行mutation的方法。

mutations：状态改变操作方法，由actions中的commit('mutation 名称')来触发。是Vuex修改state的唯一推荐方法。该方法只能进行同步操作，且方法名只能全局唯一。操作之中会有一些hook暴露出来，以进行state的监控等。

state：页面状态管理容器对象。集中存储Vue components中data对象的零散数据，全局唯一，以进行统一的状态管理。页面显示所需的数据从该对象中进行读取，利用Vue的细粒度数据响应机制来进行高效的状态更新。

getters：state对象读取方法。图中没有单独列出该模块，应该被包含在了render中，Vue Components通过该方法读取全局state对象。

## 为什么使用vuex

vuex可以实现数据响应，而sessionstorage是不可以的，我们使用vuex 的主要目的是为了各个组件之间的传参，通过数据改变视图。而sessionstorage是做不到这一点的

## nextTick是什么，什么时候调用

作用：在下一次 DOM 更新结束后执行其指定的回调。

场景：vue是异步渲染的框架，react也是，data改变之后，dom不会立刻渲染，$nextTick会在dom渲染之后被触发，以获取最新dom节点

## Vue.nextTick()

定义：在下次 DOM 更新循环结束之后执行延迟回调。在修改数据之后立即使用这个方法，获取更新后的 DOM。所以就衍生出了这个获取更新后的DOM的Vue方法。所以放在Vue.nextTick()回调函数中的执行的应该是会对DOM进行操作的 js代码；

理解：nextTick()，是将回调函数延迟在下一次dom更新数据后调用，简单的理解是：当数据更新了，在dom中渲染后，自动执行该函数，

vuex刷新丢失内容 改怎么解决这个bug

问题：F5 (163.070, 2.870, 1.79%)页面刷新，页面销毁之前的资源，重新请求，因此写在生命周期里的

vuex数据是重新初始化，无法获取的，这也就是为什么会打印出空的原因。

解决思路1：

使用Localstorage sessionStorage 或cookie

实际使用时当vuex值变化时，F5刷新页面，vuex数据重置为初始状态，所以还是要用到localStorage,

解决方法2:

插件vuex-persistedstate

vuex-persistedstate默认持久化所有state，可以指定需要持久化的state

## vue和react的区别

react是通过js来生成html，通过js才操作css，而vue是响应式的，用各自的处理方式，vue有单文件组件，可以把html、js和css写在一个文件中 react做的事情很少，很多都是交给社区去做，vue很多东西都是内置的，写起来方便一些 设计思想不同

## Vue中v-for 中的 key 设置的值

快速查找到节点，减少渲染次数，提升渲染性能

## Vue组件封装过程

● 首先，使用Vue.extend()创建一个组件

● 然后，使用Vue.component()方法注册组件

● 接着，如果子组件需要数据，可以在props中接受定义

● 最后，子组件修改好数据之后，想把数据传递给父组件，可以使用emit()方法

## 数组的哪些方法会引起视图更新 7个

push() pop() shift() unshift() splice() sort() reverse()

## Vue中的优化

**（1）代码层面的优化**

v-if 和 v-show 区分使用场景

computed 和 watch 区分使用场景

v-for 遍历必须为 item 添加 key，且避免同时使用 v-if

长列表性能优化

事件的销毁 addEventlisenter 事件监听

图片资源懒加载

路由懒加载

第三方插件的按需引入

优化无限列表性能

服务端渲染 SSR or 预渲染

**（2）Webpack 层面的优化**

Webpack 对图片进行压缩

减少 ES6 转为 ES5 的冗余代码

提取公共代码

模板预编译

提取组件的 CSS

优化 SourceMap

构建结果输出分析

Vue 项目的编译优化

**（3）基础的 Web 技术的优化**

开启 gzip 压缩

浏览器缓存

CDN 的使用

使用 Chrome Performance 查找性能瓶颈

## v-mode双绑定原理

1、v-model本质上是一个语法糖，可以看成是value + input 方法的语法糖。可以通过model的prop属性和event事件来进行自定义。

2、v-model是vue的双向绑定的指令，能将页面上控件输入的值同步更新到相关绑定的data属性， 也会在更新data绑定属性时候，更新页面上输入控件的值。

## keep-alive

keep-alive缓存vue实例，提高性能是 Vue 内置的一个组件，可以使被包含的组件保留状态，避免重新渲染 ，

提供 include 和 exclude 属性，两者都支持字符串或正则表达式，

include 表示只有名称匹配的组件会被缓存，exclude 表示任何名称匹配的

组件都不会被缓存 ，其中 exclude 的优先级比 include 高；

对应两个钩子函数 activated 和 deactivated ，当组件被激活时，触发钩子函数 activated，当组件被移除时，触发钩子函数 deactivated。

## nextTrick是什么，什么时候调用

场景：vue是异步渲染的框架，react也是，data改变之后，dom不会立刻渲染，$nextTick会在dom渲染之后被触发，以获取最新dom节点

## v-if和v-show什么区别

v-show通过css display控制显示和隐藏，v-if组件真正的渲染和销毁，而不是显示和隐藏，频繁切换状态使用v-show 否则v-if

v-if 常用于一次性改变，如根据权限决定是否显示

v-show 用于 tabs 切换

v-if 可与 templete块连用 ，v-show 不支持

## V-ON

给当前元素绑定input事件

## 说一下v-model 原理

双向数据绑定最核心的方法便是通过Object.defineProperty()来实现对属性的劫持，达到监听数据变动的目的.

先是从data里面的数据msg通过绑定到input控件和p标签上。然后input上通过v-on:input监听控件，触发change()。

调用方法都可以默认获取e事件，e.target.value是获取调用该方法的DOM对象的value值。把value值在赋给data里的msg，就是实现了双向数据绑定的原理了。

## 你说一下MVVM这些字母分别是什么意思 然后说一下你的理解、

m->model,v->view,vm->viewModel。dom通过监听事件操作vue里的data，反之vue中的data通过指令操作dom，这就是所说数据驱动视图，这就是mvvm的理解。

表单提交相关，离开页面的时候怎么实现数据的自动保存

使用vuex：将表单页面数据存入store，回来时再将所需数据取出

## 前后端联调接口的时候容易出现那些错误以及错误的诊断

1.404错误：没有路径

2、405错误

405错误表示：用来访问本页面的HTTP方法不被允许，这是因为浏览器访问网站是GET请求，后端服务器本地址只接受POST请求。

这个错误的解决方法是：把前后端的请求方式改为一致的。

3、500错误

500错误是服务器内部错误，可能是用户权限的问题导致，或者是数据库连接出现了错误。

## axios和ajax的区别

1、axios是一个基于Promise的HTTP库，而ajax是对原生XHR的封装；

2、ajax技术实现了局部数据的刷新，而axios实现了对ajax的封装。

## ajax请求的五个步骤

　　1. 创建XMLHttpRequest异步对象

　　2. 设置回调函数

　　3. 使用open方法与服务器建立连接

　　4. 向服务器发送数据

　　5. 在回调函数中针对不同的响应状态进行处理

## Localstorage、Sessionstorage、Cookie区别

1.Localstorage

　　长期存储数据，浏览器关闭数据后不丢失

2.Sessionstorage

　　数据在浏览器关闭后自动删除

3.Cookie （document.cookie）

　　网站为了标识用户身份而存储在用户本地终端（Client Side）上的数据（通常经过加密）。cookie始终在同源的http请求中携带（即使不需要）都会在浏览器和服务器端间来回传递。session storage和local storage不会自动把数据发给服务器，仅在本地保存

存储大小

　　Cookie 数据大小不会超过4K

　　Sessionstorage和Localstorage虽然也有存储大小的限制，但比cookie大得多，可以达到5M或者更多

5.有期时间

　　Localstorage 存储持久数据，浏览器关闭后数据不丢失，除非自动删除数据

　　Sessionstorage 数据在当前浏览器窗口关闭后自动删除

　　Cookie 设置的cookie过期时间之前一直有效，即使窗口或者浏览器关闭

## Vue 强制页面刷新（provide 和 inject）

路由守卫

路由守卫就是路由跳转过程中的一些钩子函数 ，在路由跳转的时候，做一些判断或其它的操作。 类似于组件生命周期钩子函数 。

1.全局路由守卫

beforeEach(to, from, next) 全局前置守卫，路由跳转前触发

beforeResolve(to, from, next) 全局解析守卫 在所有组件内守卫和异步路由组件被解析之后触发

afterEach(to, from) 全局后置守卫，路由跳转完成后触发

2.路由独享守卫

beforeEnter(to,from,next) 路由对象单个路由配置 ，单个路由进入前触发

3.组件路由守卫

beforeRouteEnter(to,from,next) 在组件生命周期beforeCreate阶段触发

beforeRouteUpdadte(to,from,next) 当前路由改变时触发

beforeRouteLeave(to,from,next) 导航离开该组件的对应路由时触发

4.参数

to： 即将要进入的目标路由对象

from： 即将要离开的路由对象

next(Function)：是否可以进入某个具体路由，或者是某个具体路由的路径

## Vue的模板编译

就是将“HTML”模板编译成render函数的过程。这个过程大致可以分成三个阶段：

解析阶段：将“HTML”模板解析成AST语法树；

优化阶段：从AST语法树中找出静态子树并进行标记（被标记的静态子树在虚拟dom比对时会被忽略，从而提高虚拟dom比对的性能）；

代码生成阶段：通过AST生成代码字符串，并最终生成render函数。

## Vue中的三种Watcher

Vue可以说存在三种watcher，第一种是在定义data函数时定义数据的render watcher；第二种是computed watcher，是computed函数在自身内部维护的一个watcher，配合其内部的属性dirty开关来决定computed的值是需要重新计算还是直接复用之前的值；第三种就是whtcher api了，就是用户自定义的export导出对象的watch属性；当然实际上他们都是通过class Watcher类来实现的。

描述

Vue.js的数据响应式，通常有以下的的场景：

数据变->使用数据的视图变。

数据变->使用数据的计算属性变->使用计算属性的视图变。

数据变->开发者主动注册的watch回调函数执行。

三个场景，对应三种watcher：

负责视图更新的render watcher。

执行计算属性更新的computed watcher。

用户注册的普通watcher api。

## Vue项目问题

### 登录中权限的设定

**两种验证机制**

登录验证方案一： 会话机制

后端提供的接口采用会话保存用户登陆状态.前端调用登录接口成功时,接口响应头通知浏览器记录会话cookie，在发起受限资源的接口时，浏览器在请求头自带改会话cookie到后端，通过这种方式判断用户是否已经登录。

会话机制方案,登录状态完全由后端维护。前端只需要在请求相应加入拦截，判断接口响应是否未登录或过期登录，如果是就重定向到登陆页面

登录验证方案二： Token机制

登录接口成功时相应token，前端收到token后保存到浏览器本地存储localStorage

访问需要权限的信息是，前端再从本地存储中读取token，携带token参数访问接口（可以在请求拦截全局处理）

前端判断接口响应是否未登录或者是登录过期，如果是，重定向到登陆页面（可在请求相应拦截全局处理）

## 路由导航守卫

vue-router 提供对应全局前置和后置回调函数，可以加入业务逻辑控制路由跳转或或取消导航。

## 请求拦截

前端框架使用 axios 类库进行http请求，支持全局 对请求发送前（request）与 响应后（response）进行注入拦截逻辑。可对发送数据与响应数据进行适配。

axios.interceptors.request.use(function (config) {

// 在这里实现对请求前的处理

return config

})

//在ajax接收响应数据之前，进行判断是否响应未登录、如果未登录重定向到登录页面

axios.interceptors.response.use(function (res) {

// 在这里实现响应后的处理

return res

})

## mounted钩子函数中请求数据导致页面闪屏问题

其实就是加载时机问题，放在created里会比mounted触发早一点，如果在页面挂载完之前请求完成的话就不会看到闪屏了

## IE9中template标签使用问题

之前在tr标签里面用template标签包裹td标签，出现了比较严重的UI错乱，

所以。。IE9不能在tr标签中使用template标签

## 封装一个简单的组件

//对input 文本框进行封装

Vue.component("tc-input", {

props: ['value', 'label'],

methods: {

updateVal: function (val) {

console.log("子控件组件：" + val);

this.$emit("input", val);

},

},

template: '<span v-if="label">{{label}}</span>

<input v-model="value" v-on:input="updateVal($event.target.value)" />'

});

## element中的上传组建 upload

问题：我在上传图片的时候，页面显示上传成功并切展示到了页面，但是在控制台Netwo (9.870, 0.000, 0.00%)rk中发现报错:出现无效的token值，既没有权限。

分析：在上传图片时，<el-upload>在发送请求时，没有使用到axios，所以没有携带token值，而是<el-upload>的内部自己封装了一套ajax，他自带的ajax是没有携带Authorization字段

解决：<el-upload>中有一个header属性用来设置上传的请求头部