目录

[Vue中如何在组件内部实现一个双向数据绑定？ 2](#_Toc34237747)

[Vue中如何监控某个属性值的变化？ 2](#_Toc34237748)

[Vue中给data中的对象属性添加一个新的属性时会发生什么，如何解决？ 2](#_Toc34237749)

[4、vue中 key 值的作用 3](#_Toc34237750)

[5、vue.js的两个核心是什么？ 3](#_Toc34237751)

[6、position的值， relative和absolute分别是相对于谁进行定位的？ 3](#_Toc34237752)

[7、cSS 选择符有哪些？哪些属性可以继承？优先级算法如何计算？ CSS3新增伪类有那些? 3](#_Toc34237753)

[伪类和伪元素区别： 4](#_Toc34237754)

[8、函数防抖和函数节流？ 4](#_Toc34237755)

[9、介绍promise 5](#_Toc34237756)

[10、axios的特点有哪些？ 5](#_Toc34237757)

[11、axios有哪些常用方法？ 5](#_Toc34237758)

[12、vue组件化的理解 5](#_Toc34237759)

[13、vuex是什么？怎么使用？哪种功能场景使用它？ 5](#_Toc34237760)

[14、请说下封装 vue 组件的过程？ 6](#_Toc34237761)

[v-show 与 v-if 区别 6](#_Toc34237762)

[动态绑定class的方法 6](#_Toc34237763)

[计算属性和 watch 的区别 6](#_Toc34237764)

[怎样理解单向数据流 7](#_Toc34237765)

[深拷贝与浅拷贝 7](#_Toc34237766)

[keep-alive 7](#_Toc34237767)

[自定义组件的语法糖 v-model 是怎样实现的 8](#_Toc34237768)

[生命周期 9](#_Toc34237769)

[组件通信 10](#_Toc34237770)

[props/$emit 10](#_Toc34237771)

[路由跳转 10](#_Toc34237772)

[vue-router 有哪几种导航钩子 10](#_Toc34237773)

[Vue.js 2.x 双向绑定原理 10](#_Toc34237774)

[什么是 MVVM，与 MVC 有什么区别 11](#_Toc34237775)

[this.$nextTick() 11](#_Toc34237776)

[理解Vue中的Render渲染函数 11](#_Toc34237777)

[vue原理 12](#_Toc34237778)

[slot插槽 12](#_Toc34237779)

# ****Vue中如何在组件内部实现一个双向数据绑定？****

父组件通过 props 传值给子组件，子组件通过 $emit 来通知父组件修改相应的props值

# ****Vue中如何监控某个属性值的变化？****

watch: {

      obj: {

      handler (newValue, oldValue) {

        console.log('obj changed')

      },

      deep: true  //深度监听

immediate:true //初次监听

    }

  }

# ****Vue中给data中的对象属性添加一个新的属性时会发生什么，如何解决？****

this.$set(this.obj, 'b', 'obj.b')  //响应式变化，值的改变会引起该对象引用的地方，以及监控生效

# 4、vue中 key 值的作用

 使用key来给每个节点做一个唯一标识

# ****5、vue.js的两个核心是什么？****

数据驱动和组件化

# 6、****position的值， relative和absolute分别是相对于谁进行定位的？****

relative:相对定位，相对于自己本身在正常文档流中的位置进行定位。

absolute:生成绝对定位，相对于最近一级定位不为static的父元素进行定位。

fixed: （老版本IE不支持）生成绝对定位，相对于浏览器窗口或者frame进行定位。

static:默认值，没有定位，元素出现在正常的文档流中。

sticky:生成粘性定位的元素，容器的位置根据正常文档流计算得出。

# 7、c****SS 选择符有哪些？哪些属性可以继承？优先级算法如何计算？ CSS3新增伪类有那些?****

选择符：

id选择器（#myId）

类选择器（.myClassName）

标签选择器（div,p,h1）

相邻选择器（h1 + p）

子选择器（ul > li）

后代选择器（li a）

通配符选择器（\*）

属性选择器（button[disabled="true"]）

伪类选择器（a:hover,li:nth-child）

优先级：

!important > 行内样式（比重1000） > id（比重100） > class/属性（比重10） > tag / 伪类（比重1）;

# 伪类和伪元素区别：

a:hover,li:nth-child

伪元素：li:before、:after,:first-letter,:first-line,:selecton

**伪类：**伪类对元素的分类是基于特征，而不是基于元素的名字属性或是内容。特征并不可以通过DOM树直接获得。

**伪类**作用的对象是整个对象

**：link**  选择所有未访问链接

**：hover**  把鼠标放在链接上的状态

**：disable**  指定当页面打开时默认处于选取状态的单选框或复选框的控件的样式

**：cehcked**   指定当表单中的radio单选框或者是checkbox复选框处于选取状态时的样式

（伪类选择器就是不存在与HTML文档中，但是用CSS语法可以选择它们，并对其能够渲染修饰）

**伪元素：**伪元素创造文档语言能够指定的文档树之外的抽象。例如：first-chlid访问第一个子元素。

**伪元素**作用于元素的一部分，一个段落的第一行或第一个字母

**::before**  在某元素之前插入某些内容

**::after**  在某元素之后插入内容

（CSS伪元素用于向某些选择器设置特殊效果）

# 8、函数防抖和函数节流？

函数防抖是指频繁触发的情况下，只有足够的空闲时间，才执行代码一次

函数防抖的要点，也是要一个setTImeout来辅助实现。延迟执行需要跑的代码

如果方法多次触发，则要把上次记录的延迟执行代码用clearTimeout清掉，重新开始

如果计时完毕，没有方法进来访问触发，则执行代码

[复制代码](javascript:void(0);)

var tiemr = false;

document.getElementById(‘debounce’,onScrll = function(){

clearTimeout(timer);

timer = setTimeout(function(){

console.log('111')

}, 300);

}

# 9、介绍promise

就是一个对象，用来传递异步操作的消息。有三种状态：pending(进行中)，resolved(已完成)和rejected(失败)

有了promise对象，就可以将异步操作以同步操作的流程表示出来，避免了层层嵌套的回调函数

10、axios的特点有哪些？  
答：  
一、Axios 是一个基于 promise 的 HTTP 库，支持promise所有的API  
二、它可以拦截请求和响应  
三、它可以转换请求数据和响应数据，并对响应回来的内容自动转换成 JSON类型的数据  
四、安全性更高，客户端支持防御 XSRF

11、axios有哪些常用方法？  
答：  
一、axios.get(url[, config])   //get请求用于列表和信息查询  
二、axios.delete(url[, config])  //删除  
三、axios.post(url[, data[, config]])  //post请求用于信息的添加  
四、axios.put(url[, data[, config]])  //更新操作

# 12、****vue组件化的理解****

组件是可复用的 Vue 实例, 如果网页中的某一个部分需要在多个场景中使用，那么我们可以将其抽出为一个组件进行复用。组件大大提高了代码的复用率

# 13、vuex是什么？怎么使用？哪种功能场景使用它？

答：vue框架中状态管理。在main.js引入store，注入。新建了一个目录store，….. export 。场景有：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

# 14、请说下封装 vue 组件的过程？

答：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。

然后，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。

# v-show 与 v-if 区别

不同点：v-if显示隐藏是将dom元素整个添加或删除，而v-show隐藏则是为该元素添加css--display:none，dom元素还在。

# 动态绑定class的方法

<div :class="{'active':isActive}"></div>

isActive: true

<div class="activeOne" v-bind:class="{ activeTwo: isActive, 'activeThree': hasError }"></div>

isActive: true,

hasError: true

<div :class="classObject"></div>

classObject: {

active: true,

}

# 计算属性和 watch 的区别

计算属性基于响应式依赖进行缓存的，计算属性的触发条件是他的依赖变化了才会重新执行

Watch则是监听某个值发生变化而触发事件

watch适合处理的场景是，侦听一个数的变化，当该数据变化，来处理其他与之相关数据的变化（该数据影响别的多个数据）  
computed适合处理的场景是，获得一个值或者结果，该结果受其他的依赖的影响。（一个数据受多个数据影响）

# 怎样理解单向数据流

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

# 深拷贝与浅拷贝

浅拷贝的理解就是，两个变量指向缓存堆（缓存栈）里的同一位置，就以为着其中一个对缓存中的数据提交改变，另外一个也会跟着改变。

深拷贝的话就是说，把原先变量指向的缓存里的数据再一模一样的复制一下，缓存中就有两个一模一样的数据，任意改变并不会引起另外一个的改变。

JSON.parse(JSON.stringify(vs))

# keep-alive

keep-alive是Vue.js的一个内置组件。它能够把不活动的组件实例保存在内存中，而不是直接将其销毁，它是一个抽象组件，不会被渲染到真实DOM中，也不会出现在父组件链中。

它提供了include与exclude两个属性，允许组件有条件地进行缓存。

keep-alive提供了两个生命钩子，分别是activated与deactivated。

实现：根据key在this.cache中查找，如果存在则说明之前已经缓存过了，直接将缓存的vnode的componentInstance（组件实例）覆盖到目前的vnode上面。否则将vnode存储在cache中。

最后返回vnode（有缓存时该vnode的componentInstance已经被替换成缓存中的了）。

# 自定义组件的语法糖 v-model 是怎样实现的

<custom

v-model='something'>

</custom>

<custom

:value="something"

@input="value => { something = value }">

</custom>

watch:{

newValue(){

this.$emit('input', this.newValue)

}

}

# 生命周期

beforeCreate () {  
 *//创建前*} ,  
created () {  
 *//创建*} ,  
beforeMount () {  
 *//渲染前  
 // this.$forceUpdate();  
 // this.$nextTick();*} ,  
mounted () {  
 *//渲染  
 alert* ( 1 );  
} ,  
activated () {  
 *//可见*} ,  
deactivated () {  
 *//隐藏*} ,  
beforeUpdate () {  
 *//更新前*} ,  
updated () {  
 *//更新*} ,  
beforeDestroy () {  
 *//销毁前*} ,  
destroyed () {  
 *//销毁*}

# 组件通信

## props/$emit

不借助其他插件的情况下使用，时效最快内存较少，代码简洁

# 路由跳转

**this.$router.push() (函数里面调用)//产生记录**

this.$router.replace() (用法同上,push)//替代，没记录

 this.$router.go(n) ()//整数负数，前进或者退后几个记录

# vue-router 有哪几种导航钩子

1、全局钩子  
2、某个路由独享的钩子  
3、组件内钩子

主要都有**三个参数**：  
to: Route: 即将要进入的路由对象  
from: Route: 当前导航正要离开的路由  
next: Function: 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(): 进行下一个钩子。如果全部钩子执行完了，则导航的状态就是 confirmed （确认的）。  
next(false): 中断当前的导航。如果浏览器的 URL 改变了（可能是用户手动或者浏览器后退按钮），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。

# Vue.js 2.x 双向绑定原理

首先我们为每个vue属性用Object.defineProperty()实现数据劫持，为每个属性分配一个订阅者集合的管理数组dep；然后在编译的时候在该属性的数组dep中添加订阅者，v-model会添加一个订阅者，{{}}也会，v-bind也会，只要用到该属性的指令理论上都会，接着为input会添加监听事件，修改值就会为该属性赋值，触发该属性的set方法，在set方法内通知订阅者数组dep，订阅者数组循环调用各订阅者的update方法更新视图。

# 什么是 MVVM，与 MVC 有什么区别

 Mvvm**定义**MVVM是Model-View-ViewModel的简写。即模型-视图-视图模型。【模型】指的是后端传递的数据。【视图】指的是所看到的页面。【视图模型】mvvm模式的核心，它是连接view和model的桥梁。它有两个方向：一是将【模型】转化成【视图】，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。二是将【视图】转化成【模型】，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。总结：在MVVM的框架下视图和模型是不能直接通信的。它们通过ViewModel来通信，ViewModel通常要实现一个observer观察者，当数据发生变化，ViewModel能够监听到数据的这种变化，然后通知到对应的视图做自动更新，而当用户操作视图，ViewModel也能监听到视图的变化，然后通知数据做改动，这实际上就实现了数据的双向绑定。并且MVVM中的View 和 ViewModel可以互相通信。

MVC**的定义：**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。MVC和MVVM的区别并不是VM完全取代了C，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。

# this.$nextTick()

是只在视图发生改变后可以进行的回调事件

# 理解Vue中的Render渲染函数

VUE一般使用template来创建HTML，然后在有的时候，我们需要使用javascript来创建html，这时候我们需要使用render函数。

**理解createElement**  
Vue通过建立一个虚拟DOM对真实的DOM发生变化保存追踪，如下代码：  
**return createElement('h1', this.title);**  
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及其子节点。我们称这样的节点为虚拟DOM，可以简写为VNode

# vue原理

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

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

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。

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。

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

# slot插槽

* 插槽（Slot）是Vue提出来的一个概念，正如名字一样，插槽用于决定将所携带的内容，插入到指定的某个位置，从而使模板分块，具有模块化的特质和更大的重用性。
* 插槽显不显示、怎样显示是由父组件来控制的，而插槽在哪里显示就由子组件来进行控制

有默认的插槽，带名字匹配的插槽