# 1.自我介绍



2分钟，300-500个字

基本信息:姓名，家乡，学校，专业，工作年限

项目介绍：最近的项目，项目周期，团队人数，我负责的功能，突出表现

意愿：岗位匹配度，未来职业规划

范文1：

我叫张三来自河南安阳林州,就读于北京农业大学软件工程专业，从事web研发工作3年，供职过两家公司，熟练掌握vue技术栈，小程序技术栈，能够使用react以及第三方UI开发框架独立自主开发项目中的一些核心功能，有良好的代码规范和封装意识，同时有良好的团队协作能力，能够有效的和后端，ui，需求等同事沟通，对新技术有强烈的追求和较强的学习能力，可以快速适应工作环境。

    最近做的一个项目是我们公司的官网新增的一个职场频道，这个项目主要是针对国内半导体领域跟集成电路企业的一些招聘难题，我主要负责的是主页面的渲染还有我的申请、收藏、评论跟简历中心的制作。

  我的人生格言是：越努力，越幸运。只有积极向上，不断进步才有可能得到自己想要的东西。我的职业规划是三年内进一步自学vue和react,在技术方面继续深耕， 争取让自己拥有良好的大局观和更广泛的视野。平时我也会多看大牛的博客日志，比如csdn,简书，博客园等。争取每天都有进步。对于贵公司我在网上也做了一些了解，我认为我和贵公司当前岗位完全匹配，完全能胜任这份工作。谢谢。

范文2：各位面试官大家好，我叫赖锦兴，来自福建省龙岩市，很荣幸今天能来贵公司进行面试。我本科毕业于福州理工学院网络工程专业，至今为止从事前端开发已经有三年了，能熟练使用 vue,react 等前端框架，对前端工程化与模块化开发有深度的理解，同时具有 Web 性能优化的实践经验。在之前工作中也开发过不少的项目，基本上是以公司自研的移动端 App 和 PC 端网站为主，偶尔根据公司需求也参与一些微信小程序和公众号的开发，慢慢地积累了比较丰富的开发经验，也逐渐形成了良好的代码规范和封装意识。

此外，在工作中我也具备良好的逻辑分析和独立解决问题的能力，即使真的碰到难题也能通过与同事们有效的沟通与团队协作，及时解决问题。

工作之余，我也会持续关注业界动态并尽可能抽出时间去看一些博客或者是个人网站，让自己对已有技术有更深层次的理解，同时学习一些前沿技术，目的在于提升自己的同时跟进技术迭代。

最后，在来之前我也对贵公司有一些了解，我认为贵公司当前所需前端岗位与我十分匹配，谢谢

# 2.vue面试题

1.v-show和v-if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 <template> 元素，也不支持 v-else

v-if 有更高的切换消耗而 v-show 有更高的初始渲染消耗

v-if 是控制元素的添加与删除，而 v-show 只是控制元素的 display 属性。

2.为何v-for要用key

快速查找到节点，减少渲染次数，提升渲染性能

3.描述vue组件声明周期

单组件声明周期图

挂载： beforeCreate => created => beforeMount => mounted

更新： beforeUpdate => updated

销毁： beforeDestroy => destroyed

父子组件生命周期图

挂载： parent beforeCreate => parent created => parent beforeMount => child beforeCreate => child created => child beforeMount => child mounted => parent mounted

更新： parent beforeUpdate => child beforeUpdate => child updated => parent updated

销毁： parent beforeDestroy => child beforeDestroy => child destroyed => parent destroyed

从以上能够看出：

挂载时，子组件是在父组件before mount后开始挂载，并且子组件先mounted，父组件随后

更新时，子组件是在父组件before update后开始更新，子组件先于父组件更新

销毁时，子组件是在父组件before destroy后开始销毁，并且是子组件先销毁，父组件随后。

4.vue组件如何通信

1.父子组件props和this.$emit

2.ref 链：父组件要给子组件传值，在子组件上定义一个 ref 属性，这样通过父组件的 $refs 属性就可以获取子组件的值了，也可以进行父子，兄弟之间的传值($parent / $children与 ref类似)

3.事件总线bus：使用一个 空的 VUE 实例作为事件总线，自定义事件event.$on event.$off event.$emit

4 provide inject组件通信

5.vuex

6.$attrs和$listeners 仅仅是传递数据，而不做中间处理，$attrs 里存放的是父组件中绑定的非 Props 属性，$listeners里存放的是父组件中绑定的非原生事件。

常见使用场景可以分为三类：

父子通信：

父向子传递数据是通过 props，子向父是通过 events（$emit）；

通过父链 / 子链也可以通信（$parent / $children）；

ref 也可以访问组件实例；

provide / inject API；

$attrs/$listeners

vuex

兄弟通信：

事件总线Bus；

Vuex

跨级通信：

事件总线Bus；

Vuex；

provide / inject API

$attrs/$listeners

5.描述组件渲染和更新的过程

1、vue 组件初次渲染过程

解析模板为 render 函数

触发响应式，监听 data 属性的 getter 和 setter

执行 render 函数， 生成 vnode，patch(elem,vnode)

2、vue 组件更新过程

修改 data， 触发 setter （此前在getter中已被监听）

重新执行 render 函数，生成 newVnode，patch(vnode, newVnode)

6.双向数据绑定v-model的实现原理

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。

7.对mvvm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。

8.computed有何特性

缓存，data不变不会重新计算，提高性能

9.VUE 中如何封装组件？什么组件，为什么要封装组件？组件中 data 为什么是一个函数？

为什么要封装组件？

　　主要就是为了解耦，提高代码复用率。

　　什么是组件？

　　页面上可以复用的都称之为组件 它是 HTML、CSS、JS 的聚合体。

　　组件就相当于库，把一些能在项目里或者不同项目里可以复用的代码进行需求性的封装。

　　组件中的 data 为什么是一个函数？

　　让每个返回的实例都可以维护一份被返回对象的独立的拷贝。

10.ajax请求应该放在哪个生命周期？

mounted，因为js是单线程，ajax异步获取数据

11.如何将组件所有props传递给子组件？

父组件绑定一个自定义属性变量，然后子组件通过props使用这个变量即可。

12.如何自定实现自定义v-model?

我们定义了model对象。model对象包含两个属性，一个是prop，一个是event。prop值text1，event的值change1，我们这里写model是为了改变默认的东西，使用我们自己定义的变量。

model:{prop:'',event:''}

<input type="text" :value="text1" @input="$emit('change1', $event.target.value)" >

13.多个组件有相同逻辑，如何抽离？

使用mixin 对公共部分的逻辑进行抽离

14.何时要使用异步组件？

加载大组件，路由异步加载

15.何时使用keep-alive？

缓存组件不需要重复渲染，多个静态tab页切换，优化性能

16.何时使用beforeDestroy？

1.解绑自定义事件event.$off

2.清除定时器

3.解绑自定义dom事件，如windom.scroll等

17.什么是作用域插槽？

在solt组件中有自己的data，把它传给使用的地方

18.vuex中action和mutation有何区别？

action中处理异步，mutation不可以

mutation做原子操作，action可以整合多个mutation

19.vue-router常用路由模式

hash默认，h5 histroy需要服务端支持

20.如何配置vue-router异步加载

component:() => import('./component')

21.请用vnode描述一个dom结构【笔试题】

<ul id='test'>

<p class='hehe'>这里是p标签</p>

<li>{{1+1}}</li>

</ul>

let vdom={

tag:'ul',

props:{

id:'test'

},

children:[

{

tag:'p',

props:{

class:'hehe'

},

children:'这里是p标签'

},

{

tag:'li',

children:2

}

]

}

22.监听data变化的核心api是什么？

vue2.0核心api是Object.defineProperty，vue3.0是启用proxy实现响应式

23.vue如何监听数据变化？

vue中的watch监听数据变化

24.请描述响应式原理？

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

监听数组变化：重写数组的push.pop.shift.unshift.splice.sort.reverse方法

25.简述diff算法过程（了解）

在执行Diff算法的过程就是调用名为 patch 的函数，比较新旧节点。一边比较一边给真实的 DOM 打补丁。patch 函数接收两个参数 oldVnode 和 Vnode，它们分别代表新的节点和之前的旧节点。这个patch函数会比较 oldVnode 和 vnode 是否是相同的, 即函数 sameVnode(oldVnode, vnode), 根据这个函数的返回结果分如下两种情况：

true：则执行 patchVnode

false：则用 vnode 替换 oldVnode

//对比过程

找到对应的真实 dom，称为 el

判断 vnode 和 oldVnode 是否指向同一个对象。

如果是，那么直接 return。

如果他们都有文本节点并且不相等，那么将 el 的文本节点设置为 vnode 的文本节点。

如果 oldVnode 有子节点而 vnode 没有，则删除 el 的子节点。

如果 oldVnode 没有子节点而 vnode 有，则将 vnode 的子节点真实化之后添加到 el

如果两者都有子节点，则执行 updateChildren 函数比较子节点。

26.vue为何是异步渲染？

因为如果不采用异步更新，那么每次更新数据都会对当前组件进行重新渲染，所以考虑性能问题，Vue会在本轮数据更新之后，再去异步更新视图

27.vue常见性能优化方式？[常见]

1.合理使用v-if和v-show，

2.合理使用computed，

3.v-for加key，

4.自定义事件，dom事件及时销毁，

5.合理使用异步组件，

6.合理使用keepalive，

7.data层级不要太深，

8.使用vue-loader在开发环境做模板编译，

9.前端通用性能优化（如图片懒加载/减少 HTTP请求数/合理设置 HTTP缓存/资源合并与压缩/合并 CSS图片/将 CSS放在 head中/避免重复的资源请求/切分到多个域名），

10.使用ssr

29、vue 的指令用法



v-html   //html

v-text   //元素里要显示的内容

v-bind：data   //绑定动态数据   ：data

v-on：click     //绑定事件       @click

v-for

v-if　　 //条件渲染指令

v-model   //双向绑定，用于表单

30、vue.js的两个核心是什么？

数据驱动和组件化

31.vue中子组件调用父组件的方法?

1.直接在子组件中通过this.$parent.event来调用父组件的方法。

2.在子组件里用$emit向父组件触发一个事件，父组件监听这个事件就行了。

3.父组件把方法传入子组件中，在子组件里直接调用这个方法。

32.vue中父组件调用子组件的方法?

父组件利用ref属性操作子组件方法。

父：

<child ref="childMethod"></child>

子：

method: {

 test() {

    alert(1)

 }

}

在父组件里调用test即 this.$refs.childMethod.test()

33.vue页面级组件之间传值?

1.使用vue-router通过跳转链接带参数传参。

2.使用本地缓存localStorge。

3.使用vuex数据管理传值

34.说说vue的动态组件。

多个组件通过同一个挂载点进行组件的切换，is的值是哪个组件的名称，那么页面就会显示哪个组件。

35.route和​router的区别是什么？

$router为VueRouter的实例，是一个全局路由对象，包含了路由跳转的方法、钩子函数等。

$route 是路由信息对象||跳转的路由对象，每一个路由都会有一个route对象，是一个局部对象，包含path,params,hash,query,fullPath,matched,name等路由信息参数。

36.为什么使用vue开发？

组件化开发 单页面路由 丰富的Api方法 双向的数据绑定 单向数据流 易于结合其他第三库

37.vue和react 有什么区别？[了解]

1、监听数据变化的实现原理不同\*\*

2、数据流的不同\*\*

3、HoC和mixins\*\*

4、组件通信的区别\*\*

5、模板渲染方式的不同\*\*

6、渲染过程不同\*\*

7、框架本质不同\*\*

38.Vuex和Redux的区别[了解]

mvvm和mvc理解 与区别？

View 将请求转交---> Controlle 处理 --->Model数据更新保存 ----->View视图显示

View 接受用户交互请求

View 将请求转交给Controller处理

Controller 操作Model进行数据更新保存

数据更新保存之后，Model会通知View更新

View 更新变化数据使用户得到反馈

MVVM即Model-View-ViewModel，将其中的 View 的状态和行为抽象化，让我们可以将UI和业务逻辑分开。MVVM的优点是低耦合、可重用性、独立开发。

View 接收用户交互请求

View 将请求转交给ViewModel

ViewModel 操作Model数据更新

Model 更新完数据，通知ViewModel数据发生变化

ViewModel 更新View数据

39.说一下钩子函数都有哪些？

钩子函数有三种 ,也叫路由守卫

全局导航钩子（跳转前进行判断拦截） 全局路由守卫

router.beforeEach(to, from, next),全局前置守卫

router.beforeResolve(to, from, next),全局的解析守卫

router.afterEach(to, from ,next) 全局的后置守卫

组件内钩子 路由独享的守卫

beforeRouteEnter

beforeRouteUpdate

beforeRouteLeave

单独路由独享组件

beforeEnter 组件内的守卫

40.双向数据绑定的理解？

vue采用数据劫持结合发布者-订阅者模式的方式，通过 Object.defineProperty劫持data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。

42.vue中哪些数组方法可以直接对数组修改实现视图更新

push() pop() shift() unshift() splice() sort() reverse() vue数组对象修改触发视图更新

43.，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。

44.v-for中key 的作用

快速查找到节点，减少渲染次数，提升渲染性能

45.使用过keep-alive吗

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。

46.computed、watch（自动监听、深度监听）、methods区别

我们可以将同一函数定义为一个 method 或者一个计算属性。对于最终的结果，两种方式是相同的。

不同点：

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。

methods：只要发生重新渲染， method 调用总会执行该函数。

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。

47.vue中对象更改检测的注意事项

由于 JavaScript 的限制，Vue 不能检测对象属性的添加或删除：

对于已经创建的实例，Vue 不能动态添加根级别的响应式属性。但是，可以使用 `Vue.set(object, key, value)`方法向嵌套对象`添加响应式属性`。为已有对象赋予多个新属性，比如使用 `Object.assign()`或 `\_.extend()`。在这种情况下，你应该用两个对象的属性创建一个新的对象。

48.什么是$nextTick？

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

49.ref 的作用？

获取dom元素 this.$refs.box

获取子组件中的data this.$refs.box.msg

调用子组件中的方法 this.$refs.box.open()

50.什么是vuex？vuex核心包括？怎么修改state中数据？在项目中哪里使用？

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 的核心包括：state（存放状态）、mutations（同步的更改状态）、actions（发送异步请求，拿到数据）、getters（根据之前的状态派发新的状态）、modules（模块划分）

　　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 里面同步的更改数据

　　应用场合：购物车的数据共享、登入注册

51.路由模式有哪些？路由传参有哪些方式？

hash模式（默认） 例如：http://abc.com/#/user/10

h5 history模式 例如：http://abc.com/user/20 需要server端支持

注意：history有如下问题 404

路由传参方法？

query传参和params传参

1、声明式导航

不带参跳转 对应的地址为/foo

url字符串拼接传参 对应的地址为/foo?id=123

query方式对象形式传参 对应的地址为/foo?id=123

params方式对象形式传参 对应地址为 /path/123 , 注意params和query一起使用params会失效，params与name一起使用

2、编程式导航(路由实例对象router=new VueRouter())

字符串router.push('home')

对象router.push({ path: 'home' })

命名的路由 对应路径为/path/123

router.push({ name: 'user', params: { userId: '123' }})

带查询参数，变成 /register?plan=123

router.push({ path: 'register', query: { plan: '123' }})

接收参数

this.$route.params.id

this.$route.query.xxx

52.vue过滤器

过滤器是将后台返回的数据换一种形式输出，不改变原来的数据 应用场景:后台返回的状态码（性别，支付状态），商品价格

1. 全局过滤器

Vue.filter('过滤器',对应的过滤器函数)

2.局部过滤器

通过在Vue实例上挂载filers添加过滤器，只能在当前组件内部使用

53.data数据结构设计思路，组件化设计原则

data数据结构设计，或者问有哪些原则

原则：【{id:1,name:'zhangsan'}】

1.用数据描述所有的内容

2.数据要结构化，易于程序操作，遍历，查找

3.数据要可扩展，以便增加新的功能

组件如何设计，有什么原则

原则：

1.从功能上拆分层次

2.尽量让组件原子化，一个组件只做一件事情

3.容器组件（只管数据，一般是顶级组件）和展示组件（只管显示视图）

54.移动端项目如何适配（rem）？

css3规定：1rem的大小就是根元素<html>的font-size的值。

通过设置 根元素<html>的font-size的大小，来控制整个html文档内的字体大小、元素宽高、内外边距等，

根据移动设备的宽度大小来实现自适应，不同的设备都展示一致的页面效果。

rem布局简单分析 分三步：

第一步：前端开发者首先拿到UI设计原型稿的宽度，750宽度，如 320px 或者640px或者750px;

第二步：增加脚本（设置根元素<html>字体大小）

第三步：css中使用rem单位；关键点：字体大小，元素宽高，内外边距一定是根据设计稿测量得来的。

55.有没有使用过axios、axios拦截器，跨域如何解决？

axios拦截器：

// 添加请求拦截器

axios.interceptors.request.use

// 添加响应拦截器

axios.interceptors.response.use

跨域特别注意两点：

第一，如果是协议和端口造成的跨域问题“前台”是无能为力的，

第二：在跨域问题上，域仅仅是通过“URL的首部”来识别而不会去尝试判断相同的ip地址对应着两个域或两个域是否在同一个ip上。

vue如何解决跨域：

proxyTable 这里vue脚手架生成的标准项目为准。一般在项目config目录下面有个index文件

CORS CORS即跨源资源共享，它定义了一种浏览器和服务器交互的方式来确定是否允许跨域请求。

Nginx 当我们明白跨越的含义之后。只要解决了'源'的问题。那么跨越也就不存在了

56.vue项目做过哪些优化

（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

57.为什么做首屏优化？

首屏时间的快与慢，直接影响到了用户对网站的认知度。所以首屏时间的长短对于用户的滞留时间的长短、用户转化率都尤为重要。

58.如何做首屏优化？

css模块化加载，对应模块下的css交给js或jsonp请求返回

js懒执行，有交互才执行

图片在其他屏（非首屏）都采用懒加载的模式，这样既能节省流量，也能减少请求数或延迟请求数。

服务器方面：

1. 少量静态文件的域名，图片与iconfont均是放在同一个域下，减少DNS的解析事件，最好做到域名收敛。

2. 模块化接口的支持。

3. 首屏内容最好做到静态缓存

4. 对于vue和react做ssr

59.vue常用的修饰符

.stop - 调用 event.stopPropagation()，禁止事件冒泡。

.prevent - 调用 event.preventDefault()，阻止事件默认行为。

.capture - 添加事件侦听器时使用 capture 模式。

.self - 只当事件是从侦听器绑定的元素本身触发时才触发回调。

.native - 监听组件根元素的原生事件。

.once - 只触发一次回调。

v-model 指令常用修饰符：

.number - 输入字符串转为数字

.trim - 输入首尾空格过滤

.lazy

60.v-on可以监听多个方法吗？

v-on可以监听多个方法，但是同一种事件类型的方法，vue-cli工程会报错

64.v-for产生的列表，实现active的切换 tab切换

v-for生成序列

<ul>

<li v-for="(info,index) in list" :key="info.id" @click="select(index)" v-bind:class="{'active':info.active}">{{info.name}}</li>

</ul>

data数据

list:[

{

name:'a',

id:1,

active:false

},

{

name:'b',

id:2,

active:false

},

{

name:'c',

id:3,

active:false

},

{

name:'d',

id:4,

active:false

},

]

点击事件

select(d){

this.list.map(s=>s.active=false); //for forEach map filter some

this.list[d].active=true;

},

CSS样式

<style scoped>

li.active{

background-color: red;

}

65.v-model语法糖使用

v-model语法糖 v-model其实是一种简写方式，我们常见的有两种v-model，分别是input元素上的v-model 和非input元素上v-model

input元素上的：

<input v-model="price"><!-- 下行的语法糖 -->

<input :value="price" @input="price = $event.target.value">

data(){

return {

price: 20

}

}

非input元素上的：

Vue.component('base-checkbox', {

model: {

prop: 'checked',

event: 'change'

},

template: `

<input

type="checkbox"

v-bind:checked="checked"

v-on:change="$emit('change', $event.target.checked)"

>

`

})

父组件：<base-checkbox v-model="lovingVue"></base-checkbox>

66.十个常用的自定义过滤器

（1）去除空格

type:1-所有空格 2-前后空格 3-前空格 4-后空格。

（2）任意格式日期处理

（3）字母大小写切换

type:1:首字母大写 2：首页母小写 3：大小写转换 4：全部大写 5：全部小写

（4）字符串循环复制,count->次数.

（5）字符串替换

（6）字符替换\*，隐藏手机号或者身份证号等

（7）格式化处理字符串

（8）现金额大写转换函数

（9）保留2位小数 0.3 + 0.9 ！= 1.2

（10）补零

67.vue等单页面应用及其优缺点

单页面应用（SPA），通俗一点说就是指只有一个主页面的应用

单页面的优点：

用户体验好，快，内容的改变不需要重新加载整个页面，基于这一点spa对服务器压力较小。

前后端分离。

效果会比较炫酷（比如切换页面内容时的专场动画）。

单页面缺点：

不利于seo。

导航不可用，如果一定要导航需要自行实现前进、后退。（由于是单页面不能用浏览器的前进后退功能，所以需要自己建立堆栈管理）。

初次加载时耗时多。

页面复杂度提高很多。

68.vue的计算属性，特性，应用

含义：computed 是计算属性，把模板中的一些稍微复杂的逻辑计算放回到js代码中，解决在模板中放入太多的逻辑会让模板过重且难以维护的问题。

computed特性：

（1）computed 是基于它们的依赖进行缓存的

（2）只有在它的相关依赖发生改变时才会重新求值

computed应用：就是简化tempalte里面{{}}计算和处理props或$emit的传值

69.vue父组件向子组件通过props传递数据

父组件传递：

<子组件调用 v-bind:自定义属性名="要传递的数据"></子组件调用>

子组件接收：

props:['自定义属性名']

props:{

type:String,

default{

return ''

}

}

72.vue-cli中自定义指令的使用

1.全局注册

Vue.directive(‘name’, {})

2.局部注册

directives: {

name: {}

}

然后在模版中直接使用即可。

73.父组件异步获取动态数据传递给子组件(好题)

问题：由于父组件中的数据是异步获取的，而子组件在一开始便会渲染，所以会造成子组件渲染完成后，数据还未获取到的情况

解决方案：在子组件渲染前，判断父组件数据是否获取完成，数据获取完成后再渲染子组件.

//tab-weekly(v-if="userId", :userId="userId")

//tab-weekly是子组件，userId是在父组件中异步获取、需要传递给子组件tab-weekly的数据，在其中加一个判断，//当userId存在后，再渲染子组件

74.父组件给子组件props传参，子组件接收的6种方法

1. data中 变量 = this.props里面的数据

2. watch监听 赋值

3. mounted 渲染完成后调用一个函数 进行赋值

4. vuex

5. computed 计算属性，用法和watch类似，computed是同步，watch可以异步

6. 父组件v-if 触发渲染和销毁，子组件触发传参

75.Vuex页面刷新数据丢失咋解决这个bug

问题：F5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

76.按钮权限怎么做？

在点击左侧菜单，存储全部权限，每次点击单个时候，去计算获取当前页面的按钮权限，封装一个button组件，然后在需要的地方引用

77.完整的说下从url解析到显示页面过程，结合项目中说

1. 首先浏览器主进程接管，开了一个下载线程。

2. 然后进行HTTP请求（DNS查询、IP寻址等等），中间会有三次捂手，等待响应，开始下载响应报文。

3. 将下载完的内容转交给Renderer进程管理。

4. Renderer进程开始解析css rule tree和dom tree，这两个过程是并行的，所以一般我会把link标签放在页面顶部。

5. 解析绘制过程中，当浏览器遇到link标签或者script、img等标签，浏览器会去下载这些内容，遇到时候缓存的使用缓存，不适用缓存的重新下载资源。

6. css rule tree和dom tree生成完了之后，开始合成render tree，这个时候浏览器会进行layout，开始计算每一个节点的位置，然后进行绘制。

7. 绘制结束后，关闭TCP连接，过程有四次挥手

78.vue声明周期都在哪些场景中使用？

1.beforeCreate(){}

　　创建前，访问不到data当中的属性以及methods当中的属性和方法，可以在当前生命周期创建一个loading，在页面加载完成之后将loading移除

2.created(){}

　　创建后，当前生命周期执行的时候会遍历data中所有的属性，给每一个属性都添加一个getter、setter方法,将data中的属性变成一个响应式属性

3. beforeMount(){}

　　 模板与数据进行结合，但是还没有挂载到页面上。因此我们可以在当前生命周期中进行数据最后的修改

4.mounted(){}

　　当前生命周期数据和模板进行相结合，并且已经挂载到页面上了，因此我们可以在当前生命周期中获取到真实的DOM元素

5. beforeUpdate(){}

　　当数据发生改变的时候当前生命周期就会执行，因此我们可以通过当前生命周期来检测数据的变化

　　当前生命周期执行的时候会将更新的数据与模板进行相结合，但是并没有挂载到页面上，因此我们可以在当前生命周期中做更新数据的最后修改

6.updated(){}

　　数据与模板进行相结合，并且将更新后的数据挂载到了页面上。因此我们可以在当前生命周期中获取到最新的DOM结构

7. beforeDestroy(){}

　　当前生命周期中我们需要做事件的解绑 监听的移除 定时器的清除等操作

8. destroyed(){}

　　当前生命周期执行完毕后会将vue与页面之间的关联进行断开

当<keep-alive>包裹动态组件的时候会触发两个新的生命周期

9.　　activated 当组件为活跃状态的时候触发(活跃状态：进入页面的时候)

10. deactivated 缓存状态的时候触发

# 3.html、css面试题

#### 1.html5有哪些新特性

H5新特性有：

语义标签、增强型表单、Canvas绘图、地理定位、SVG绘图、拖放API、WebWorker、WebStorage、WebSocket

#### 2.什么是重绘和回流？两者区别？

回流：

当渲染树中的一部分(或全部)因为元素的规模尺寸，布局，隐藏等改变而需要重新构建。这就称为回流(reflow)。

width

height

padding

border

margin

position

top

left

bottom

right

float

clear

text-align

vertical-align

line-height

font-weight

font-size

font-family

overflow

white-space

重绘：

当渲染树中的一些元素需要更新属性，而这些属性只是影响元素的外观，风格，而不会影响布局的，则就称为重绘。

color

border-style

border-radius

text-decoration

box-shadow

background

两者区别：

回流必将引起重绘，而重绘不一定会引起回流。比如：只有颜色改变的时候就只会发生重绘而不会引起回流

当页面布局和几何属性改变时就需要回流

#### 3.CSS 选择符有哪些？优先级算法如何计算？

1.id选择器（#id）

2.类选择器（.class）

3.标签选择器（div，h1，p）

4.相邻选择器（h1 + p）

5.子选择器（ul > li）

6.后代选择器（li a）

7.通配符选择器（ \* ）

8.属性选择器（a[title]）

9.伪类选择器（a:hover，li:nth-child）

**优先级算法如何计算？**

1.优先级就近原则，同权重情况下样式定义最近者为准；

2.载入样式以最后载入的定位为准；

3.!important > id > class > tag；

4.important 比 内联优先级高，但内联比id要高；

1、第一等：代表内联样式，如: style=””，权值为1000。

2、第二等：代表ID选择器，如：#content，权值为0100。

3、第三等：代表类，伪类和属性选择器，如.content，权值为0010。

4、第四等：代表类型选择器和伪元素选择器，如div p，权值为0001。

5、通配符、子选择器、相邻选择器等的。如\*、>、+,权值为0000。

6、继承的样式没有权值。

#### 4.CSS3有哪些新特性？

word-wrap 文字换行

text-overflow 超过指定容器的边界时如何显示

text-decoration 文字渲染

text-shadow文字阴影

gradient渐变效果

transition过渡效果 transition-duration：过渡的持续时间

transform拉伸，压缩，旋转，偏移等变换

animation动画

#### 5.谈谈你对盒模型的认识

专业的面试，一定会问 CSS 盒模型。对于这个题目，我们要回答一下几个方面：

（1）基本概念：content、padding、margin。

（2）标准盒模型、IE盒模型的区别。不要漏说了IE盒模型，通过这个问题，可以筛选一部分人。

（3）CSS如何设置这两种模型（即：如何设置某个盒子为其中一个模型）？如果回答了上面的第二条，还会继续追问这一条。

（4）JS如何设置、获取盒模型对应的宽和高？这一步，已经有很多人答不上来了。

（5）实例题：根据盒模型解释\*\*边距重叠\*\*。

前四个方面是逐渐递增，第五个方面，却鲜有人知。

（6）BFC（边距重叠解决方案）或IFC。

注意：

浮动元素和绝对定位元素不会发生边界坍塌

只有块级元素的垂直方向才存在margin合并的问题，再说一下margin负值向内部缩减，正值向外。

#### 6.简述flex布局【重要】

flex弹性布局，可以简便、完整、响应式地实现各种页面布局, 容器默认存在两根轴：水平的主轴（main axis）和垂直的交叉轴（cross axis）

flex-direction：决定主轴的方向

row（默认值）：主轴为水平方向，起点在左端。

row-reverse：主轴为水平方向，起点在右端。

column：主轴为垂直方向，起点在上沿。

column-reverse：主轴为垂直方向，起点在下沿。

flex-wrap：换行

nowrap（默认）：不换行。

wrap：换行，第一行在上方。

wrap-reverse：换行，第一行在下方。

flex-flow：flex-flow属性是flex-direction属性和flex-wrap属性的简写形式，默认值为row nowrap。

justify-content：主轴对齐方式

flex-start（默认值）：左对齐

flex-end：右对齐

center： 居中

space-between：两端对齐，项目之间的间隔都相等。

space-around：每个项目两侧的间隔相等。所以，项目之间的间隔比项目与边框的间隔大一倍。

align-items：侧轴对齐方式

flex-start：交叉轴的起点对齐。

flex-end：交叉轴的终点对齐。

center：交叉轴的中点对齐。

baseline: 项目的第一行文字的基线对齐。

stretch（默认值）：如果项目未设置高度或设为auto，将占满整个容器的高度。

#### 7.css 隐藏元素有哪几种方法？

css隐藏元素方法

1.display:none

2.Jquery：show(),hide(),toggle()方法就是通过改变display的值来实现变化效果的。

3.visibility:hidden：

和display:none的区别在于，元素在页面消失后，占据的空间依旧会保留着，但是display:none不会，所以它只会导致浏览器重绘而不会回流，因此，visibility:hidden适用于那些元素隐藏后不希望页面布局发生变化的场景；

4.opacity:0 (透明度，元素本身不会隐藏，会引起重绘)

5.overflow:hidden

6.position: absolute： 把元素脱离文档流移出视觉区域，既不会影响布局，又能让元素保持可以操作。应用该属性后，主要就是通过控制方向（top,left,right,bottom），达到一定的值，离开当前页面。

clip-path（不常用）

#### 8.BFC（Block Formatting Context） 是什么？应用？

是一个独立的渲染区域，通俗点BFC就是页面上的一个隔离的独立容器，容器里面的元素布局不受外界影响，同时也不会影响到外面的元素。

那么我们可以通过CSS为元素设置一些属性，来触发BFC,常用的方式有：

Float值不为none

Postion值不为relative和static

Overflow值为auto scroll和hidden

display值为inline-block

BFC用途：

1、防止margin重叠2、清楚内部浮动3、消除文本环绕做出自适应两栏布局

#### 9.清除浮动的方法（4种）

浮动为该元素脱离文档流，不占据空间。浮动元素碰到包含它的边框或者浮动元素的边框停留。

清除浮动的方法：



1, 添加额外标签

在浮动元素末尾添加一个空的标签，

<div style="clear:both"></div>

优点：通俗易懂，容易掌握

缺点：可以想象通过此方法，会添加多少无意义的空标签，有违结构与表现的分离，在后期维护中将是噩梦。

2, 使用 br标签和其自身的 html属性

<br clear="all" />

优点：比空标签方式语义稍强，代码量较少

缺点：同样有违结构与表现的分离，不推荐使用

3, 父元素设置 overflow：hidden

优点：不存在结构和语义化问题，代码量极少

缺点：内容增多时候容易造成不会自动换行导致内容被隐藏掉，无法显示需要溢出的元素；overflow:hidden会导致中键失效。

4、父元素设置 overflow：auto 属性。同样IE6需要触发hasLayout，演示和3差不多

优点：不存在结构和语义化问题，代码量极少

缺点：多个嵌套后，firefox某些情况会造成内容全选；IE中 mouseover 造成宽度改变时会出现最外层模块有滚动条等，firefox早期版本会无故产生focus等。

5,使用:after 伪元素

需要注意的是 :after是伪元素（Pseudo-Element），不是伪类（某些CSS手册里面称之为“伪对象”），很多清除浮动大全之类的文章都称之为伪类，不过csser要严谨一点，这是一种态度。由于IE6-7不支持:after，使用 zoom:1触发 hasLayout。

.clearfix:after {content:" "; display:block; visibility:hidden; clear:both; }

.clearfix { \*zoom:1; }

优点：结构和语义化完全正确,代码量居中

缺点：复用方式不当会造成代码量增加

#### 10.****哪些外边距不重叠的情况？****

1）水平margin永远不会重合

2）设置了overflow属性(visible除外)的元素和它的子元素之间的margin不会重叠

3）设置了绝对定位（position:absolute）的盒模型，垂直margin不会被重叠，和子元素之间也不重叠

4）设置了display:inline-block的元素，垂直margin不会重叠，和子元素之间也不重叠

5）根元素(如HTML)与body的margin不会重叠

#### 11.如何让一个盒子水平垂直居中【重要】

①定位：（常用方法,推荐）

position:absolute;

position:relative;

left:值;

top:值;

②display:table-cell;

③外边距：margin-left:值;

margin-top:值;

④margin:auto;（不兼容低版本的IE浏览器）

⑤弹性盒模型：display:flex;（常用方法,推荐）

justify-content:conter;

align-itens:center;

⑥用transform的属性translate平移，不仅能实现绝对居中同样的效果，也支持联合可变高度方式使用。

使用top:50%; left:50%;时，是以盒子的左上角为原点定位，是左上角的原点居中，但是元素本身并不居中。

transform：translate(-50%，-50%):分别向左向上移动自身长宽的50%，使其位于中心。

transform: translate(-50%,-50%)导致的像素模糊问题解决办法：

/\*\* 这 0.5px加或者减都可以 \*/

transform: translat(calc(-50% + 0.5 px), calc(-50% + 0.5 px))；

⑦用calc计算

# 4.js面试题

##### 1.JS数据类型

1基本数据类型:string,number,Boolean,null,undefined;

2引用数据类型object(Object,Array),function

3 ES6新增 symbol

##### 2.JS判断数据类型的方法

typeof检测基本数据类型

instanceof检测引用数据类型

Object.prototype.toString.call()精准判断数据类型

##### 3.null和undefined区别

Undefined类型只有一个值，即undefined。当声明的变量还未被初始化时，变量的默认值为undefined。

用法：

变量被声明了，但没有赋值时，就等于undefined。

调用函数时，应该提供的参数没有提供，该参数等于undefined。

对象没有赋值的属性，该属性的值为undefined。

函数没有返回值时，默认返回undefined。

Null类型也只有一个值，即null。null用来表示尚未存在的对象，常用来表示函数企图返回一个不存在的对象。

用法：

作为函数的参数，表示该函数的参数不是对象。

作为对象原型链的终点。

##### 4.JS作用域的理解

JS中的作用域分为两种：全局作用域和函数作用域。

函数作用域中定义的变量，只能在函数中调用，外界无法访问。没有块级作用域导致了if或for这样的逻辑语句中定义的变量可以被外界访问，因此ES6中新增了let和const命令来进行块级作用域的声明。

##### 5.call,apply和bind区别

三个函数的作用都是将函数绑定到上下文中，用来改变函数中this的指向；三者的不同点在于语法的不同。

fun.call(thisArg[, arg1[, arg2[, ...]]])

fun.apply(thisArg, [argsArray])

所以 apply和 call的区别是 call方法接受的是若干个参数列表，而 apply接收的是一个包含多个参数的数组。

而bind()方法创建一个新的函数, 当被调用时，将其this关键字设置为提供的值，在调用新函数时，在任何提供之前提供一个给定的参数序列。

var bindFn = fun.bind(thisArg[, arg1[, arg2[, ...]]])

bindFn()

##### 6.this指向

1.普通函数，定时器，立即执行

this -> window

2.obj.fun1()

this -> obj

3. var obj = new Object()

this -> obj

4. <button onclick="fun3(this)">点击</button>

this -> button[绑定的对象]

5.fun3.call({name:'liujie'})

this -> {name:'liujie'}

6.es6 箭头函数的this 继承外层函数调用的 this

##### 7.深拷贝和浅拷贝

深拷贝和浅拷贝最根本的区别在于是否真正获取一个对象的复制实体，而不是引用。

浅复制：仅仅是指向被复制的内存地址，如果原地址发生改变，那么浅复制出来的对象也会相应的改变。

深复制：在计算机中开辟一块新的内存地址用于存放复制的对象。

##### 8.谈谈防抖和节流

防抖节流就是使用定时器来实现我们的目的。

防抖(debounce)：

在事件被触发n秒后再执行回调，如果在这n秒内又被触发，则重新计时。

典型的案例就是输入框搜索：输入结束后n秒才进行搜索请求，n秒内又输入的内容，则重新计时。

节流(throttle)：

规定在一个单位时间内，只能触发一次函数，如果这个单位时间内触发多次函数，只有一次生效。

典型的案例就是鼠标不断点击触发，规定在n秒内多次点击只生效一次。

为什么要掌握防抖和节流

函数节流与函数防抖都是可以限制函数的执行频次，根据不同的场景来对执行频率进行限制，避免了函数触发频率过高导致的响应速度跟不上触发频率，出现延迟，假死或卡顿的现象。

##### 9.谈谈cookie,sessionStorage和localStorage

相同点：

cookie、sessionStorage和localStorage都用在客户端存储数据，每一个都有自己的存储和到期限制

不同点：

一、存储大小

cookie数据大小不能大于4K；

localStorage和sessionStorage则可以达到5M；

二、有效时间

cookie在设置的有效期内一直有效；

localStorage存储持久数据，只要不手动清除则一直存在；

sessionStorage数据在当前浏览器关闭后就会被自动清除

三、数据与服务器间的交互方式

cookie的数据会自动传递到服务器端，服务器端也可以写cookie到客户端

localStorage和sessionStorage不会把数据自动传到服务器端，仅在本地存储

​

##### 10.0.1+0.2!=0.3怎么处理

在计算机中数字无论是定点数还是浮点数都是以多位二进制的方式进行存储的,计算机天生只能存储整数，它只能用某种方法来表示小数

解决方法：

1.把计算数字 提升 10 的N次方 倍 再 除以 10的N次方。N>1.

(0.1\*10+ 0.2\*10)/10== 0.3//true

2.四舍五入 tofixed

##### 11.数组的常用方法(至少说出8个)

1.shift 删除数组中的第一个元素

2.pop 删除数组中的最后一个元素

3.unshift 增加元素在数组的前面

4.push 增加元素在数组的后面

5.map 循环，并且返回新的数组

6.forEach 循环，遍历

7.filter 过滤，筛选出数组中的满足条件的，并且返回新的数组

8.concnt 合并数组

9.find 查找出第一个符合条件中的数组元素

10.findIndex 查找出第一个符合条件中的数组元素，所在的索引位置

11.flat 将多维数组转为一维数组

12.join将数组转为字符串

13.reverse 颠倒数组中的顺序

14.every检测数组中元素是否都是符合条件 === bollean

15.some检测数组中元素是否有满足条件的元素 === bollean

16.splice(start,n,添加元素) 开始位置 删除个数，添加元素

17.sort 排序

18.slice(start,end) 选中[start.end)之间的元素

19.indexOf 查找值所在的位置

20.includes 查看数组中是否存在此元素

##### 12.new一个对象的过程中发生了什么吗?

1. 创建空对象；

var obj = {};

2. 设置新对象的constructor属性为构造函数的名称，设置新对象的\_\_proto\_\_属性指向构造函数的prototype对象；

obj.\_\_proto\_\_ = ClassA.prototype;

3. 使用新对象调用函数，函数中的this被指向新实例对象：

ClassA.call(obj);//{}.构造函数();

4. 将初始化完毕的新对象地址，保存到等号左边的变量中

**13.JS实现继承(至少会一种)**

1. 原型链继承

2. 借用构造函数继承

3. 组合继承继承

4. 寄生式继承继承

6. 原型式继承继承

##### 14.get和post区别

| **请求方式** | **GET** | **POST** |
| --- | --- | --- |
| 参数位置 | 参数拼接到url的后面 | 参数在请求体中 |
| 参数大小 | 受限于浏览器url大小，一般不超过32K | 1G |
| 服务器数据接收 | 接收1次 | 根据数据大小，可分多次接收 |
| 适用场景 | 从服务器端获取数据 | 向服务器提交数据 |
| 安全性 | 参数携带在url中，安全性低 | 相对于GET请求，安全性更高 |

##### 15.JSONP原理【了解】

由于浏览器的同源策略限制，不允许跨域请求；但是页面中的 script、img、iframe标签是例外，不受同源策略限制。

Jsonp 就是利用 script标签跨域特性进行请求。

JSONP 的原理就是，先在全局注册一个回调函数，定义回调数据的处理；与服务端约定好一个同名回调函数名，服务端接收到请求后，将返回一段 Javascript，在这段 Javascript 代码中调用了约定好的回调函数，并且将数据作为参数进行传递。当网页接收到这段 Javascript 代码后，就会执行这个回调函数。

JSONP缺点：它只支持 GET请求，而不支持 POST请求等其他类型的HTTP请求。

##### 16.缓存的理解

缓存分为强缓存和协商缓存。

强缓存不过服务器，协商缓存需要过服务器，协商缓存返回的状态码是304。两类缓存机制可以同时存在，强缓存的优先级高于协商缓存。当执行强缓存时，如若缓存命中，则直接使用缓存数据库中的数据，不再进行缓存协商。

强缓存:

Expires http1.0

cache-control http1.1

协商缓存:

Last-Modified 和 If-Modified-Since http1.0

Etag 和 If-None-Match http1.1

##### 17.XSS和CSRF区别【了解】

跨站脚本攻击（Cross Site Scripting)，为了不和层叠样式表 CSS 混淆，故将跨站脚本攻击缩写为 XSS。恶意攻击者往 Web 页面里插入恶意 Script 代码，当用户浏览该页之时，嵌入其中 Web 里面的 Script 代码会被执行，从而达到恶意攻击用户的目的。

跨站请求伪造（Cross-site request forgery），是伪造请求，冒充用户在站内的正常操作。我们知道，绝大多数网站是通过 cookie 等方式辨识用户身份，再予以授权的。所以要伪造用户的正常操作，最好的方法是通过 XSS 或链接欺骗等途径，让用户在本机（即拥有身份 cookie 的浏览器端）发起用户所不知道的请求。

区别：

原理不同，CSRF是利用网站A本身的漏洞，去请求网站A的api；XSS是向目标网站注入JS代码，然后执行JS里的代码。

CSRF需要用户先登录目标网站获取cookie，而XSS不需要登录

CSRF的目标是用户，XSS的目标是服务器

XSS是利用合法用户获取其信息，而CSRF是伪造成合法用户发起请求

##### 18.HTTP与HTTPS的区别

HTTP的URL由 http://起始且默认使用端口80，而HTTPS的URL由 https://起始且默认使用端口443

HTTP是超文本传输协议，信息是明文传输，HTTPS则是具有安全性的 SSL 加密传输协议

HTTP的连接很简单，是无状态的，HTTPS 协议是由 SSL+HTTP 协议构建的可进行加密传输、身份认证的网络协议，比 http 协议安全

##### 19.HTTP状态码

1XX 信息性状态码

2XX 成功状态码

3XX 重定向状态码

4XX客户端错误状态码

5XX服务器错误状态码

我们常见的状态码有200--表示客户端请求成功，400--表示客户端请求有语法错误，403--表示服务器收到请求，但是没有权限，404--表示请求资源不存在，500--表示服务器端错误

##### 20.事件捕获、事件冒泡、阻止事件冒泡、阻止默认事件

事件捕获

捕获型事件(event capturing)：事件从最不精确的对象(document 对象)开始触发，然后到最精确(也可以在窗口级别捕获事件，不过必须由开发人员特别指定)

事件冒泡

冒泡型事件：事件按照从最特定的事件目标到最不特定的事件目标(document对象)的顺序触发。

阻止事件的传播,阻止事件冒泡：

• 在W3c中，使用stopPropagation（）方法

• 在IE下设置cancelBubble = true；

阻止事件的默认行为，例如click <a>后的跳转~

• 在W3c中，使用preventDefault（）方法；

• 在IE下设置window.event.returnValue = false;

##### 21.Js事件循环机制

在线程之内，又被分为了两个队列：同步任务队列,异步任务队列

同步环境执行(step1) -> 事件循环1(step4) -> 事件循环2(step4的重复)…

　　其中的异步进程有：

　　　　a、类似onclick等，由浏览器内核的DOM binding模块处理，事件触发时，回调函数添加到任务队列中；

　　　　b、setTimeout等，由浏览器内核的Timer模块处理，时间到达时，回调函数添加到任务队列中；

　　　　c、Ajax，由浏览器内核的Network模块处理，网络请求返回后，添加到任务队列中。

异步执行的运行机制

（1）所有同步任务都在主线程上执行，形成一个执行栈（execution context stack）。

（2）主线程之外，还存在一个"任务队列"（task queue）。只要异步任务有了运行结果，就在"任务队列"之中放置一个事件。

（3）一旦"执行栈"中的所有同步任务执行完毕，系统就会读取"任务队列"，看看里面有哪些事件。那些对应的异步任务，于是结束等待状态，进入执行栈，开始执行。

（4）主线程不断重复上面的第三步。

##### 22.什么是闭包，使用场景及优缺点？

闭包的概念

当一个函数内部嵌套另一个函数，子函数可以访问到父级函数中的变量，就是闭包。严格来说，闭包产生的三个条件

1.必须是嵌套关系 2.内部函数访问其所在的作用域 3.在所在作用域外被调用

闭包的作用与应用场景

作用一 使变量在内存中保存不被销毁。

作用二，延长了函数内局部变量的作用范围。

##### 23.箭头函数和普通函数的区别？

（1）箭头函数体内的this对象，就是定义时所在的对象，而不是使用时所在的对象。

（2）箭头函数不可以当作构造函数，也就是说，不可以使用new命令，否则会抛出一个错误。

（3）箭头函数不可以使用arguments对象，该对象在函数体内不存在。如果要用，可以用 rest 参数代替。

（4）箭头函数不可以使用yield命令，因此箭头函数不能用作 Generator 函数

（5）箭头函数使用call()和apply()调用

##### 24.什么是promise?

简单来说可以把promise当作一个装着异步操作结果的容器。从语法上说，Promise 是一个对象，从它可以获取异步操作的消息。Promise 提供统一的 API，各种异步操作都可以用同样的方法进行处理。它将异步函数以同步的方式书写，也解决了回调地狱问题

特点：（1）对象状态不受外界影响

（2）一旦状态改变，就不会再变，任何时候都可以得到这个结果

缺点：（1），无法取消promise，一旦新建它就会立即执行，无法中途取消

（2），如果不设置回调函数，promise内部抛出的错误，不会反应到外部

（3）无法得知目前进展到哪一个阶段（刚刚开始还是即将结束）

三个状态：进行中、已成功、以失败。

##### 25.什么是ajax？

即异步的 JavaScript 和 XML，是一种用于创建快速动态网页的技术；传统的网页（不使用 AJAX）如果需要更新内容，必需重载整个网页面。使用AJAX则不需要加载更新整个网页，实现部分内容更新。

# 5.兼容性面试题

#### 1、Css3 的新属性？

当一些 CSS3 样式语法还存在波动时，它们提供针对浏览器的前缀。现在主要流行的浏览器内核主要有：Trident 内核：主要代表为 IE 浏览器Gecko 内核：主要代表为 Firefox FFPresto 内核：主要代表为 OperaWebkit 内核：产要代表为 Chrome 和 Safari而这些不同内核的浏览器，CSS3 属性（部分需要添加前缀的属性）对应需要添加不同的前缀，也将其称之为浏览器的私有前缀，添加上私有前缀之后的 CSS3 属性可以说是对应浏览器的私有属性：Trident 内核：前缀为-msGecko 内核：前缀为-mozPresto 内核：前缀为-oWebkit 内核：前缀为-webkit

#### 2、window.event || event

问题：使用 window.event 无法在 FF 上运行解决方法：FF 的 event 只能在事件发生的现场使用，此问题暂无法解决。可以把 event 传到函数里变通解决：onMouseMove = "functionName(event)"function functionName (e) {e = e || window.event;......}

#### 移动端常见的兼容性问题

##### 1.html5调用安卓或者ios的拨号功能

html5提供了自动调用拨号的标签，只要在a标签的href中添加tel：就可以了。如下：< a href=" ">400-810-6999 转 1034</ a>拨打手机如下：< a href="tel:15677776767">点击拨打 15677776767 </ a>

##### 2.上下拉动滚动条时卡顿、慢

body {

-webkit-overflow-scrolling: touch;

overflow-scrolling: touch;

}

Android3+和 iOS5+支持 CSS3 的新属性为 overflow-scrolling。

##### 3.圆角bug

某些 Android 手机圆角失效background-clip: padding-box;

##### 4.ios 设置input 按钮样式会被默认样式覆盖

解决方式如下：

input,textarea {border: 0;-webkit-appearance: none;}设置默认样式为 none

##### 5.IOS键盘字母输入，默认首字母大写

解决方案，设置如下属性

<input type="text"autocapitalize="off"/>

##### 6.h5底部输入框被键盘遮挡问题

h5页面有个问题是，当输入框在最底部，点击软键盘后输入框会被遮挡。

解决办法：由于弹起输入法，会执行onresize 事件，根据窗口变化，将原先是固定定位的元素改为position:static;。当关闭输入法时再切换回position:absolute;。

var getHeight = $(document).height();

$(window).resize(function(){

if($(document).height() < getHeight) {

$('#footer').css('position','static');

}else {

$('#footer').css('position','absolute');

}

});

##### 7.IOS移动端click事件300ms的延迟响应

解决方案：

1、fastclick可以解决在手机上点击事件的300ms延迟

2、zepto的touch模块，tap事件也是为了解决在click的延迟问题

3、触摸事件的响应顺序为touchstart --> touchmove --> touchend --> click,也可以通过绑定ontouchstart事件，加快对事件的响应，解决300ms延迟问题

##### 8.在ios和andriod中,audio元素和video元素在无法自动播放

应对方案：触屏即播$('html').one('touchstart',function(){audio.play()})

##### 9.CSS动画页面闪白,动画卡顿

解决方法:

1.尽可能地使用合成属性transform和opacity来设计CSS3动画，不使用position的left和top来定位

2.开启硬件加速

-webkit-transform: translate3d(0, 0, 0);

-moz-transform: translate3d(0, 0, 0);

-ms-transform: translate3d(0, 0, 0);

transform: translate3d(0, 0, 0);

##### 10.fixed定位缺陷

iOS 下，fixed定位的元素里如果有input，键盘弹起后，点击定位会上移，显示定位也会出问题

解决：不使用fixed

body {height: 100%;}

使用absolute 替代 fixed，相对于body定位

11.1px问题

1、 媒体查询利用设备像素比缩放，设置小数像素；

.border { border: 1px solid #999 }

@media screen and (-webkit-min-device-pixel-ratio: 2) {

.border { border: 0.5px solid #999 }

}

@media screen and (-webkit-min-device-pixel-ratio: 3) {

.border { border: 0.333333px solid #999 }

}

2.transform: scale(0.5) 方案

div {

height:1px;

background:#000;

-webkit-transform: scaleY(0.5);

-webkit-transform-origin:0 0;

overflow: hidden;

}

12.首屏白屏问题

减少入口文件

静态资源本地存储

图片压缩

UI框架按需加载

重复组件打包优化

SSR

GZip

13.轮播图提前占位

height:0;

font-size:0;

padding-bottom:33%

14.滚动条隐藏

在PC端隐藏html右侧默认滚动条

html {

/\*隐藏滚动条，当IE下溢出，仍然可以滚动\*/

-ms-overflow-style:none;

/\*火狐下隐藏滚动条\*/

scrollbar-width: none;

}

/\*Chrome下隐藏滚动条，溢出可以透明滚动\*/

html::-webkit-scrollbar{width:0px}

移动端隐藏滚动条

1）给滚动条的部分设置宽高为100%， overflow-y: auto;

2）设置滚动条的部分::-webkit-scrollbar{

width: 0;

display:none;

}

# 6.vue3.0面试题

##### 1.Vue3.0和Vue2.0的区别

· 3.0比2.0 快2倍

· 3.0去掉了filter, 么有beforeCreate created,用setup取代

· reactivity是可以单独作为库使用的

· 单独功能可以抽离 取代了mixin  优于mixin 解决上下反复横跳

· 支持多个子节点 fragment

· setup里没有this

· Proxy实现响应式不需要set delete  兼容性并不好

· 响应式方面 性能得到很大提升 不用初始化的时候就递归遍历属性

· 响应式不区分数组和对象

· 3.0兼容IE12以上

· composition api 可以和 options API 同时存在

##### 2.Vue3.0都有哪些重要新特性？

建议往`Composition API`和`Tree-shaking`方面答，对应比较`React Hooks`和webpack 的`Tree-shaking`

composition-api解决了什么问题

1.使用传统的option配置方法写组件的时候问题，随着业务复杂度越来越高，代码量会不断的加大；由于相关业务的代码需要遵循option的配置写到特定的区域，导致后续维护非常的复杂，同时代码可复用性不高，而composition-api就是为了解决这个问题而生的

compositon-api提供了一下几个函数

reactive

watchEffect

computed

ref

toRefs

生命周期的hooks

2.Vue3提出万物皆可 TreeShaking的概念

什么是 TreeShaking

TreeShaking 是一个术语，指的是在打包构建过程中移除没有被引用到的代码，这些代码可以成为 dead code。这个概念最早在基于 ES6 的打包工具 Rollup 中提出，后来被引入到 webpack 中。TreeShaking 比较依赖于 ES6 模块系统的静态结构特性，比如 import 和 export。

##### 3.Vue3.0对比Vue2.0的优势在哪？

vue3.0 的发布与 vue2.0 相比，优势主要体现在：更快、更小、更易维护、更易于原生、让开发者更轻松；

更快

　　1、virtual DOM 完全重写，mounting & patching 提速 100%；

　　2、更多编译时 （compile-time）提醒以减少 runtime 开销；

　　3、基于 Proxy 观察者机制以满足全语言覆盖以及更好的性能；

　　4、放弃 Object.defineProperty ，使用更快的原生 Proxy；

　　5、组件实例初始化速度提高 100%;

　　6、提速一倍/内存使用降低一半；

更小

　　1、Tree-shaking 更友好；

　　2、新的 core runtime：~ 10kb gzipped

# 7.移动端、小程序面试题

#### 1. 移动端兼容适配

<meta name="viewport" content="width=device-width, initial-scale=1.0">

适配方案：

1.rem, em, 百分比

2.框架的栅格布局

3.media query媒体查询

4.手淘团队的一套flexible.js, 自动判断dpr进行整个布局视口的放缩

#### 2。为什么以iPhone6为标准的设计稿的尺寸是以750px宽度来设计的呢？

iPhone6的满屏宽度是375px，而iPhone6采用的视网膜屏的物理像素是满屏宽度的2倍，也就是dpr(设备像素比)为2, 并且设计师所用的PS设计软件分辨率和像素关系是1:1。所以为了做出的清晰的页面，设计师一般给出750px的设计图，我们再根据需求对元素的尺寸设计和压缩。

#### 3.移动端首屏优化

采用服务器渲染ssr

按需加载配合webpack分块打包, 通过entry和commonChunkPlugin

很有必要将script标签➕异步

有轮播图 最好给个默认 另外要处理图片懒加载

打包线上也要注意去掉map 文件

组件, 路由懒加载

webpack的一切配置 肯定是必须的

压缩图片 https://tinypng.com/ 建议还是用webpack的图片压缩插件

骨架屏

Loading页面

#### 4. 自适应和响应式布局的区别

1. 自适应布局通过检测视口分辨率，来判断当前访问的设备是：pc端、平板、手机，从而请求服务层，返回不同的页面；响应式布局通过检测视口分辨率，针对不同客户端在客户端做代码处理，来展现不同的布局和内容。

2. 自适应布局需要开发多套界面，而响应式布局只需要开发一套界面就可以了。

3. 自适应对页面做的屏幕适配是在一定范围：比如pc端一般要大于1024像素，手机端要小于768像素。而响应式布局是一套页面全部适应。

4. 自适应布局如果屏幕太小会发生内容过于拥挤。而响应式布局正是为了解决这个问题而衍生出的概念，它可以自动识别屏幕宽度并做出相应调整的网页设计。

#### 5.你是怎么封装微信小程序的数据请求的？

1、 将所有的接口放在统一的 js 文件中并导出

2、在 app.js 中创建封装请求数据的方法

3、在子页面中调用封装的方法请求数据

#### 6.小程序有哪些参数传值的方法？

1、给 HTML 元素添加 data-\*属性来传递我们需要的值，然后通过 e.currentTarget.dataset 或 onload 的

param 参数获取。但 data-名称不能有大写字母和不可以存放对象

2、设置 id 的方法标识来传值通过 e.currentTarget.id 获取设置的 id 的值,然后通过设置全局对象的方式来

传递数值

3、在 navigator 中添加参传值

#### 7.webview 中的页面怎么跳回小程序中？

<web-view/>网页中可使用 JSSDK 1.3.2 提供的接口返回小程序页面。

例如：wx.miniProgram.navigateTo({url: '/path/to/page'})

#### 8.小程序关联微信公众号如何确定用户的唯一性？

使用 wx.getUserInfo 方法 withCredentials 为 true 时可获取 encryptedData，里面有 union\_id。后端需要进行对称解密。

#### 9.小程序如何实现下拉刷新？

用 view 代替 scroll-view,,设置 onPullDownRefresh 函数实现。

1、在 json 文件中配置 enablePullDownRefresh 为 true(app.json 中在 window 中设置 enablePullDownRefresh,

此效果作用于全局)。

2、在 js 文件中实现 onPullDownRefresh 方法,在网络请求完成后调用 wx.stopPullDownRefresh()来结束下拉刷

新。

#### 10.小程序调用后台接口遇到哪些问题？

1、数据的大小有限制，超过范围会直接导致整个小程序崩溃，除非重启小程序；

2、小程序不可以直接渲染文章内容页这类型的 html 文本内容，若需显示要借住插件，但插件渲染会导致

页面加载变慢，所以最好在后台对文章内容的 html 进行过滤，后台直接处理批量替换 p 标签 div 标签为 view 标

签，然后其它的标签让插件来做，减轻前端的时间。

页面加载变慢，所以最好在后台对文章内容的 html 进行过滤，后台直接处理批量替换 p 标签 div 标签为 view 标

签，然后其它的标签让插件来做，减轻前端的时间。

#### 11.小程序的 wxss 和 css 有哪些不一样的地方？

1、wxss 的图片引入需使用外链地址。

2、没有 Body，样式可直接使用 import 导入。

#### 12.分析下微信小程序的优劣势

优 势：

1、无需下载，通过搜索和扫一扫就可以打开。

2、良好的用户体验：打开速度快。

3、开发成本要比 App 要低

4、安卓上可以添加到桌面，与原生 App 差不多。

5、为用户提供良好的安全保障。小程序的发布，微信拥有一套严格的审查流程， 不能通过审查的小程序是无法发布到线上的。

劣势：

1、限制较多。页面大小不能超过 1M。不能打开超过 5 个层级的页面。

2、样式单一。小程序的部分组件已经是成型的了，样式不可以修改。例如：幻灯片、导航。

3、推广面窄，不能分享朋友圈，只能通过分享给朋友，附近小程序推广。其中附近小程序也受到微信的限制。

4、依托于微信，无法开发后台管理功能。

# 8.es6面试题

## ****1、var、let、const之间的区别****

var声明变量可以重复声明，而let不可以重复声明var是不受限于块级的，而let是受限于块级var会与window相映射（会挂一个属性），而let不与window相映射var可以在声明的上面访问变量，而let有暂存死区，在声明的上面访问变量会报错const声明之后必须赋值，否则会报错const定义不可变的量，改变了就会报错const和let一样不会与window相映射、支持块级作用域、在声明的上面访问变量会报错

## ****2、使用箭头函数应注意什么？****

（1）用了箭头函数，this就不是指向window，而是父级（指向是可变的）（2）不能够使用arguments对象（3）不能用作构造函数，这就是说不能够使用new命令，否则会抛出一个错误（4）不可以使用yield命令，因此箭头函数不能用作 Generator 函数

## ****3、ES6的模板字符串有哪些新特性？并实现一个类模板字符串的功能****

基本的字符串格式化。将表达式嵌入字符串中进行拼接。用${}来界定在ES5时我们通过反斜杠()来做多行字符串或者字符串一行行拼接。ES6反引号(``)就能解决类模板字符串的功能



let name = 'web';

let age = 10;

let str = '你好，${name} 已经 ${age}岁了'

str = str.replace(/\$\{([^}]\*)\}/g,function(){

    return eval(arguments[1]);

  })

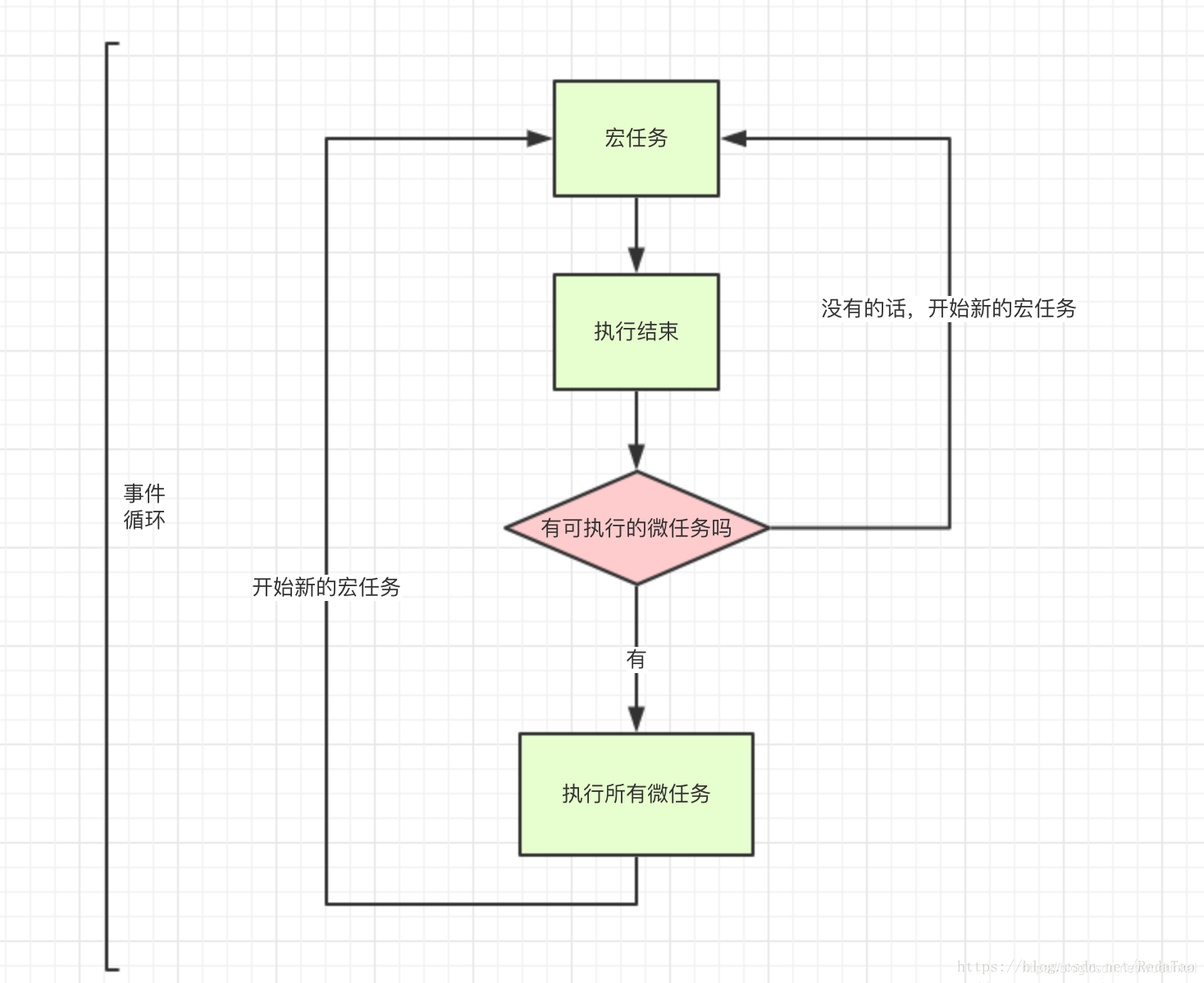
console.log(str);//你好，web 已经 10岁了

## ****4、Promise构造函数是同步执行还是异步执行，那么 then 方法呢？****

promise构造函数是同步执行的，then方法是异步执行的

## ****5、setTimeout、Promise、Async/Await 的区别****

<https://blog.csdn.net/u012925833/article/details/89306184>



事件循环中分为宏任务队列和微任务队列

宏任务：Node中fs可以进行异步的I/O操作 setTimeout setInterval 事件绑定 ajax

微任务：Promise.then catch finally

其中setTimeout的回调函数放到宏任务队列里，等到执行栈清空以后执行promise.then里的回调函数会放到相应宏任务的微任务队列里，等宏任务里面的同步代码执行完再执行async函数表示函数里面可能会有异步方法，await后面跟一个表达式async方法执行时，遇到await会立即执行表达式，然后把表达式后面的代码放到微任务队列里，让出执行栈让同步代码先执行

setTimeOut函数为异步任务，for循环为同步任务，setTimeOut里的函数为回调函数。执行顺序为：同步优先，异步靠边，回调垫底

## ****6、Promise 中reject 和 catch 处理上有什么区别****

reject 是用来抛出异常，catch 是用来处理异常reject 是 Promise 的方法，而 catch 是 Promise 实例的方法reject后的东西，一定会进入then中的第二个回调，如果then中没有写第二个回调，则进入catch网络异常（比如断网），会直接进入catch而不会进入then的第二个回调

## ****7、forEach、for in、for of三者区别****

forEach更多的用来遍历数组for in 一般常用来遍历对象或jsonfor of数组对象都可以遍历，遍历对象需要通过和Object.keys()for in循环出的是key，for of循环出的是value

# 9.React面试题

react组件如何通信

父组件向子组件通信：使用 props

子组件向父组件通信：使用 props 回调

跨级组件间通信：使用 context 对象

非嵌套组件间通信：使用事件订阅

jsx本质是什么

jsx其实是语法糖，开发环境会将jsx编译成js代码，jsx的写法大大降低了学习成本和编码工作量

context是什么，有何用途

在react应用中，数据总是通过 props 自上而下进行传递。 这种做法对于某些类型的属性而言是极其繁琐的（例如：地区偏好，UI 主题）。**Context 可以共享对于一个组件树而言是“全局”的数据。这样就不必显式地通过组件树的逐层传递 props**

setState是同步还是异步？

既可能是同步的,也可能是异步的。 准确地说,在React内部机制能检测到的地方, setState就是异步的;在React检测不到的地方,例如setInterval,setTimeout里,setState就是同步更新的

什么是纯函数

一个函数的返回结果只依赖其参数，并且执行过程中没有副作用。

react发起ajax应该在哪个生命周期

componentDidMount生命周期中调用

渲染列表，为何使用key？

key是给每一个虚拟dom的唯一id,可以依靠key,更准确, 更快的拿到旧虚拟dom中对应的新虚拟dom节点。

函数组件和class组件区别

函数组件纯函数，输入props，输出jsx

函数组件没有实例，没有生命周期，没有state

函数组件不能扩展其他方法

class创建的组件,有自己的私有数据(this.state)和生命周期

什么是受控组件

简单的说就是input里面的值受state值控制

何时使用异步组件

加载大组件 路由异步加载 资源异步加载

多个组件有公共逻辑，如何抽离

HOC，高阶组件 高阶组件不是一种功能，而是一种模式

Render Props核心思想：通过一个函数将class组件的state作为props传递给纯函数组件

redux如何进行异步请求 ajax请求

action异步函数中发送ajax请求

react-router如何配置懒加载

lazyload-loader方式 import Shop from lazy（()=>import('./src/view/Shop')）';

箭头函数方式 component:resolve => require(['@/component/Login'],resolve)·

PureComponent和Component有何区别

React.PureComponent 与 React.Component 几乎完全相同，但 React.PureComponent 通过props和state的浅对比来实现 shouldComponentUpate()。

在PureComponent中，如果包含比较复杂的数据结构，可能会因深层的数据不一致而产生错误的否定判断，导致界面得不到更新。

react事件和dom事件区别

1. event 是 SyntheticEvent 组合事件对象，模拟出来 DOM 事件所有能力

2. event.nativeEvent 是原生事件对象

3. 所有的事件，都被挂载到 document 上，为了性能

4. 和 DOM 事件不一样，和 Vue 事件也不一样

react性能优化

使用shouldComponentUpdate来优化组件

shouldComponentUpdate的实现方式，shouldComponentUpdate在比对prop和上次渲染所用的prop方面，依然用的是尽量简单的方法，做的是所谓的“浅层比较”。简单来说就是用JavaScript的===操作符来比较，如果prop的类型是字符串或者数字，只要值相同，那么“浅层比较”也会认为二者相同，但是，如果prop的类型是复杂对象，那么“浅层比较”的方式只看这两个prop是不是同一个对象的引用，如果不是，哪怕这两个对象中的内容完全一样，也会被认为是两个不同的prop。

react和vue区别

1、vue是响应式的数据双向绑定系统，而react是单向数据流，没有双向绑定。

2、vue的语法较为简单，适用于小型项目创建，而react更适用于Web端和原生App的开发，侧重于大型应用。

3、vue拥有更快的渲染速度和更小的体积，react则拥有更大的生态圈，可以带来更多的工具支持。

4、模板渲染方式的不同

在表层上，模板的语法不同，React是通过JSX渲染模板。而Vue是通过一种拓展的HTML语法进行渲染，但其实这只是表面现象，毕竟React并不必须依赖JSX。

在深层上，模板的原理不同，这才是他们的本质区别：React是在组件JS代码中，通过原生JS实现模板中的常见语法，比如插值，条件，循环等，都是通过JS语法实现的，更加纯粹更加原生。而Vue是在和组件JS代码分离的单独的模板中，通过指令来实现的，比如条件语句就需要 v-if 来实现对这一点，这样的做法显得有些独特，会把HTML弄得很乱。

21.react vdom和vue vdom区别

react和vue他们两个在渲染虚拟dom的过程其实有点相互模仿的。

当render函数渲染完毕后。他们共同使用了diff算法，来更新虚拟dom。

区别：

vue 利用双端优化,vue3有最长递增子序列

react 利用fiber,时间切片，将树变为链表

23调用 setState 之后发生了什么？

在代码中调用 setState 函数之后，React 会将传入的参数对象与组件当前的状态合并，然后触发所谓的调和过程（Reconciliation）。经过调和过程，React 会以相对高效的方式根据新的状态构建 React 元素树并且着手重新渲染整个 UI 界面。在 React 得到元素树之后，React 会自动计算出新的树与老树的节点差异，然后根据差异对界面进行最小化重渲染。在差异计算算法中，React 能够相对精确地知道哪些位置发生了改变以及应该如何改变，这就保证了按需更新，而不是全部重新渲染。

24react 生命周期函数

初始化阶段：

getDefaultProps:获取实例的默认属性

getInitialState:获取每个实例的初始化状态

componentWillMount：组件即将被装载、渲染到页面上

render:组件在这里生成虚拟的 DOM 节点

componentDidMount:组件真正在被装载之后

运行中状态：

componentWillReceiveProps:组件将要接收到属性的时候调用

shouldComponentUpdate:组件接受到新属性或者新状态的时候（可以返回 false，接收数据后不更新，阻止 render 调用，后面的函数不会被继续执行了）

componentWillUpdate:组件即将更新不能修改属性和状态

render:组件重新描绘

componentDidUpdate:组件已经更新

销毁阶段：

componentWillUnmount:组件即将销毁

25.shouldComponentUpdate 是做什么的，（react 性能优化是哪个周期函数？）

shouldComponentUpdate 这个方法用来判断是否需要调用 render 方法重新描绘 dom。因为 dom 的描绘非常消耗性能，如果我们能在 shouldComponentUpdate 方法中能够写出更优化的 dom diff 算法，可以极大的提高性能。

26.为什么虚拟 dom 会提高性能?(必考)

虚拟 dom 相当于在 js 和真实 dom 中间加了一个缓存，利用 dom diff 算法避免了没有必要的 dom 操作，从而提高性能。

用 JavaScript 对象结构表示 DOM 树的结构；然后用这个树构建一个真正的 DOM 树，插到文档当中当状态变更的时候，重新构造一棵新的对象树。然后用新的树和旧的树进行比较，记录两棵树差异把 2 所记录的差异应用到步骤 1 所构建的真正的 DOM 树上，视图就更新了。

27.react diff 原理（常考，大厂必考）

把树形结构按照层级分解，只比较同级元素。

给列表结构的每个单元添加唯一的 key 属性，方便比较。

React 只会匹配相同 class 的 component（这里面的 class 指的是组件的名字）

合并操作，调用 component 的 setState 方法的时候, React 将其标记为 dirty.到每一个事件循环结束, React 检查所有标记 dirty 的 component 重新绘制.

选择性子树渲染。开发人员可以重写 shouldComponentUpdate 提高 diff 的性能。

28.React 中 refs 的作用是什么？

Refs 是 React 提供给我们的安全访问 DOM 元素或者某个组件实例的句柄。

29.展示组件(Presentational component)和容器组件(Container component)之间有何不同

展示组件关心组件看起来是什么。展示专门通过 props 接受数据和回调，并且几乎不会有自身的状态，但当展示组件拥有自身的状态时，通常也只关心 UI 状态而不是数据的状态。

容器组件则更关心组件是如何运作的。容器组件会为展示组件或者其它容器组件提供数据和行为(behavior)，它们会调用 Flux actions，并将其作为回调提供给展示组件。容器组件经常是有状态的，因为它们是(其它组件的)数据源。

30.类组件(Class component)和函数式组件(Functional component)之间有何不同

类组件不仅允许你使用更多额外的功能，如组件自身的状态和生命周期钩子，也能使组件直接访问 store 并维持状态

当组件仅是接收 props，并将组件自身渲染到页面时，该组件就是一个 '无状态组件(stateless component)'，可以使用一个纯函数来创建这样的组件。这种组件也被称为哑组件(dumb components)或展示组件

31.(组件的)状态(state)和属性(props)之间有何不同

State 是一种数据结构，用于组件挂载时所需数据的默认值。State 可能会随着时间的推移而发生突变，但多数时候是作为用户事件行为的结果。

Props(properties 的简写)则是组件的配置。props 由父组件传递给子组件，并且就子组件而言，props 是不可变的(immutable)。组件不能改变自身的 props，但是可以把其子组件的 props 放在一起(统一管理)。Props 也不仅仅是数据--回调函数也可以通过 props 传递。

33.何为高阶组件(higher order component)

高阶组件是一个以组件为参数并返回一个新组件的函数。HOC 运行你重用代码、逻辑和引导抽象。最常见的可能是 Redux 的 connect 函数。除了简单分享工具库和简单的组合，HOC 最好的方式是共享 React 组件之间的行为。如果你发现你在不同的地方写了大量代码来做同一件事时，就应该考虑将代码重构为可重用的 HOC。

34.为什么建议传递给 setState 的参数是一个 callback 而不是一个对象

因为 this.props 和 this.state 的更新可能是异步的，不能依赖它们的值去计算下一个 state。

33除了在构造函数中绑定 this，还有其它方式吗

你可以使用属性初始值设定项(property initializers)来正确绑定回调，create-react-app 也是默认支持的。在回调中你可以使用箭头函数，但问题是每次组件渲染时都会创建一个新的回调。

35.(在构造函数中)调用 super(props) 的目的是什么

在 super() 被调用之前，子类是不能使用 this 的，在 ES2015 中，子类必须在 constructor 中调用 super()。传递 props 给 super() 的原因则是便于(在子类中)能在 constructor 访问 this.props。

36.应该在 React 组件的何处发起 Ajax 请求

在 React 组件中，应该在 componentDidMount 中发起网络请求。这个方法会在组件第一次“挂载”(被添加到 DOM)时执行，在组件的生命周期中仅会执行一次。更重要的是，你不能保证在组件挂载之前 Ajax 请求已经完成，如果是这样，也就意味着你将尝试在一个未挂载的组件上调用 setState，这将不起作用。在 componentDidMount 中发起网络请求将保证这有一个组件可以更新了。

37.描述事件在 React 中的处理方式。

为了解决跨浏览器兼容性问题，您的 React 中的事件处理程序将传递 SyntheticEvent 的实例，它是 React 的浏览器本机事件的跨浏览器包装器。

这些 SyntheticEvent 与您习惯的原生事件具有相同的接口，除了它们在所有浏览器中都兼容。有趣的是，React 实际上并没有将事件附加到子节点本身。React 将使用单个事件监听器监听顶层的所有事件。这对于性能是有好处的，这也意味着在更新 DOM 时，React 不需要担心跟踪事件监听器。

39.React 中有三种构建组件的方式

​ React.createClass()、ES6 class 和无状态函数。

40.react 组件的划分业务组件技术组件？

根据组件的职责通常把组件分为 UI 组件和容器组件。

UI 组件负责 UI 的呈现，容器组件负责管理数据和逻辑。

两者通过 React-Redux 提供 connect 方法联系起来。

41React 项目用过什么脚手架（本题是开放性题目）

creat-react-app Yeoman 等

42了解 redux 么，说一下 redux 把

redux 是一个应用数据流框架，主要是解决了组件间状态共享的问题，原理是集中式管理，主要有三个核心方法，action，store，reducer，工作流程是 view 调用 store 的 dispatch 接收 action 传入 store，reducer 进行 state 操作，view 通过 store 提供的 getState 获取最新的数据，flux 也是用来进行数据操作的，有四个组成部分 action，dispatch，view，store，工作流程是 view 发出一个 action，派发器接收 action，让 store 进行数据更新，更新完成以后 store 发出 change，view 接受 change 更新视图。Redux 和 Flux 很像。主要区别在于 Flux 有多个可以改变应用状态的 store，在 Flux 中 dispatcher 被用来传递数据到注册的回调事件，但是在 redux 中只能定义一个可更新状态的 store，redux 把 store 和 Dispatcher 合并,结构更加简单清晰

43redux 有什么缺点

一个组件所需要的数据，必须由父组件传过来，而不能像 flux 中直接从 store 取。

当一个组件相关数据更新时，即使父组件不需要用到这个组件，父组件还是会重新 render，可能会有效率影响，或者需要写复杂的 shouldComponentUpdate 进行判断。

44.setState在哪个生命周期不能用？

setState 是一个异步方法，一个生命周期内所有的 setState 方法会合并操作

componentWillMount 里 setState 会和 初始化 state 合并执行，因此这是无意义的 setState componentDidMount 中 setState 会导致渲染两遍，应该尽量避免。除了获取DOM元素位置或者宽高等 componentWillUnmount 中 setState 不会更新 state，是不生效而且无意义的 shouldComponentUpdate 或 componentWillUpdate 里调用 setState 会再次触发这两个函数，然后在两个函数又触发了 setState，死循环.. componentDidUpdate 执行 setState 渲染两遍，如非必须，尽量避免 componentWillReceiveProps 中 setState，不会造成二次渲染，可以放心setState

# 10.webpack面试题

### 1. webpack与grunt、gulp的不同？

三者都是前端构建工具，grunt和gulp在早期比较流行，现在webpack相对来说比较主流，不过一些轻量化的任务还是会用gulp来处理，比如单独打包CSS文件等。

grunt和gulp是基于任务和流（Task、Stream）的。类似jQuery，找到一个（或一类）文件，对其做一系列链式操作，更新流上的数据， 整条链式操作构成了一个任务，多个任务就构成了整个web的构建流程。

webpack是基于入口的。webpack会自动地递归解析入口所需要加载的所有资源文件，然后用不同的Loader来处理不同的文件，用Plugin来扩展webpack功能。

### 2. 与webpack类似的工具还有哪些？谈谈你为什么最终选择（或放弃）使用webpack？

同样是基于入口的打包工具还有以下几个主流的：

webpack

rollup

parcel

**从应用场景上来看：**

webpack适用于大型复杂的前端站点构建

rollup适用于基础库的打包，如vue、react

parcel适用于简单的实验性项目，他可以满足低门槛的快速看到效果

由于parcel在打包过程中给出的调试信息十分有限，所以一旦打包出错难以调试，所以不建议复杂的项目使用parcel

### 3.有哪些常见的Loader？他们是解决什么问题的？

file-loader：把文件输出到一个文件夹中，在代码中通过相对 URL 去引用输出的文件

url-loader：和 file-loader 类似，但是能在文件很小的情况下以 base64 的方式把文件内容注入到代码中去

source-map-loader：加载额外的 Source Map 文件，以方便断点调试

image-loader：加载并且压缩图片文件

babel-loader：把 ES6 转换成 ES5

css-loader：加载 CSS，支持模块化、压缩、文件导入等特性

style-loader：把 CSS 代码注入到 JavaScript 中，通过 DOM 操作去加载 CSS。

eslint-loader：通过 ESLint 检查 JavaScript 代码

### 4.有哪些常见的Plugin？他们是解决什么问题的？

html-webpack-plugin：可以复制一个有结构的html文件，并自动引入打包输出的所有资源（JS/CSS）clean-webpack-plugin：重新打包自动清空 dist 目录mini-css-extract-plugin：提取 js 中的 css 成单独文件optimize-css-assets-webpack-plugin：压缩cssuglifyjs-webpack-plugin：压缩jscommons-chunk-plugin：提取公共代码

### 5.Loader和Plugin的不同？

**不同的作用**

**Loader**直译为"加载器"。Webpack将一切文件视为模块，但是webpack原生是只能解析js文件，如果想将其他文件也打包的话，就会用到loader。 所以Loader的作用是让webpack拥有了加载和解析非JavaScript文件的能力。

**Plugin**直译为"插件"。Plugin可以扩展webpack的功能，让webpack具有更多的灵活性。 在 Webpack 运行的生命周期中会广播出许多事件，Plugin 可以监听这些事件，在合适的时机通过 Webpack 提供的 API 改变输出结果。

**不同的用法**

**Loader**在module.rules中配置，也就是说他作为模块的解析规则而存在。 类型为数组，每一项都是一个Object，里面描述了对于什么类型的文件（test），使用什么加载(loader)和使用的参数（options）

**Plugin**在plugins中单独配置。 类型为数组，每一项是一个plugin的实例，参数都通过构造函数传入。

### 6.如何利用webpack来优化前端性能？（提高性能和体验）

用webpack优化前端性能是指优化webpack的输出结果，让打包的最终结果在浏览器运行快速高效。

压缩代码。删除多余的代码、注释、简化代码的写法等等方式。可以利用webpack的UglifyJsPlugin和ParallelUglifyPlugin来压缩JS文件， 利用cssnano（css-loader?minimize）来压缩css

利用[CDN](https://cloud.tencent.com/product/cdn?from=10680)加速。在构建过程中，将引用的静态资源路径修改为CDN上对应的路径。可以利用webpack对于output参数和各loader的publicPath参数来修改资源路径

删除死代码（Tree Shaking）。将代码中永远不会走到的片段删除掉。可以通过在启动webpack时追加参数--optimize-minimize来实现

提取公共代码。

# 12.非技术面试题收集

**1.如何讲解：项目所有角色，项目全流程，各阶段常见问题**

PM产品经理 UE UI视觉设计师 PE前端开发 RD后端开发 CRD移动端开发 QA测试人员



需求经理拿到需求，并确认需求，然后开组会讲解需求并分配需求，如果有难点的技术功能实现需大家商讨下如何实现。待大家都能了解需求和功能实现之后进行开发，2周一个迭代，其中包括修改上次迭代提出的bug，并更新发布最新版进行复测，待所有功能实现完毕后，经需求经理确认功能，ui确认设计稿还原度后然后经由测试组进行测试，测试通过之后进行上线部署；

需求分析--技术方案设计-- 开发--联调--测试--上线

**2.、请问你了解过我们公司吗？**

我查看了关于贵公司的信息，知道你们近几年发展势头在加大，根据客观环境的变化，贵公司高层有意逐步改变发展战略和竞争战略，而且准备拓展国际市场，与海外销售商合作。

**3、你最大的缺点是什么？**

我经常在一件事上死磕到底，不完成誓不罢休。（有坚持的品质）

**4、在业余时间你通常喜欢干什么？有什么兴趣爱好？**

没事的时候，我会学习茶艺、插花，因为我想了解更多、学习更多，目的就是不断提升自己。我也会经常出去旅行，感受旅行的意义，目前已经去了10多个城市。

**5、你觉得你个性上最大的优点是什么？**

沉着冷静、条理清楚、立场坚定、顽强向上、乐于助人和关心他人、适应能力和幽默感、乐观和友爱。我经过一到两年的培训及项目实战，加上实习工作，使我适合这份工作。

**6、你对薪资的要求？**

我对工资没有硬性要求，我相信贵公司在处理我的问题上会友善合理。我注重的是找对工作机会，所以只要条件公平，我则不会计较太多。

**7.可以独立开发吗？**

还行 ，自己能独立开发，时间排版，以及实现的功能

**8**.**使用vue做过几个项目？**

3个项目 ，两个PC端，一个移动端，能够熟练运用vue进行项目开发

**9.外派岗位接受吗**

根据公司的需要，如果我需要外派或出差，我会毫不犹豫接受。这是我的荣幸，因为我能在不同的地方学到 更多知识、累积更多经验。

**10.期望薪资是多少？**

我对工资没有硬性要求，我相信贵公司在处理我的问题上会友善合理。我注重的是找对工作机会，所以只要条件公平，我则不会计较太多。

**11.说下你上家公司？**

公司名称【重要】

公司地址【重要】

标志建筑物

交通【重要】

项目描述

我的职责

印象深刻的事

离职原因【重要】

开发中遇到的问题，如何解决【重要】

团队人员-某一个同事 技术经理 hr

**12.背调需填表格**

公司名称

部门名称

项目名称

离职原因

薪资

工作态度

入职离职日期

我在当前公司的职务

# 13.项目项话术

建造师全题库——移动端 App项目描述:建造师全题库 App 是北京天龟教育科技有限公司旗下一款建造师真题模考刷题软件,是一款以使用 React 框架开发的学习教育类型移动端 App,其开发背景是为了服务广大的建造师考试人员,具备视频课以及直播课堂功能,还有在线搜题、模拟考试等功能。项目职责:1、负责选课以及课程详情页面的开发；2、负责发现页面倒计时功能模块实现；3、联调接口获取真实题库数据,以及题目搜索功能实现；4、负责我的页面各部分功能模块、以及个人资料数据上传。技术要点:1、利用 create-react-app 进行项目基础搭建和模块化开发。2、使用 customize-cra 修改 webpack 部分配置；2、使用 antd-mobile 组件库实现部分功能的快速开发；3、利用 react、react-thunk、react-redux 管理各组件之间的数据通讯；4、利用 Hooks 更好的连接组件，复用组件逻辑。

对应B简历话术：

我在项目中主要负责的是首页模块跟发现页模块，在首页模块中我们会把用户一周的做题数据展示出来方便给用户更加直观的对自己能力水平的评估，然后就是章节练习、真题模拟、错题收集、收藏记录的几个功能。在也封装了两个组件，一个是基于 canvas 实现的一个类似签名效果的功能，主要用于用户在做题的时候可以在屏幕上做草稿，然后还有一个就是题目纠错组件，用于就是在用户对题目解析或者是答案有疑惑时提交错误的。其次在首页模块中还放入了部分直播课跟视频课，提供给用户试听。然后我还负责发现模块的开发，主要功能是一个倒计时计算，用户输入一个日期，计算出当前日期到考试日期的天数，还是比较简单，然后是用户能力评估功能，主要是通过出一些不同类型的题目来给用户做，通过得分来判断用户各个类型知识的能力。还有一个题目搜索功能就是把用户输入的题目关键字传给后端再把相关题目返回。项目人员配置：

1 个需求经理，3 个前端，4 个后端，1 个 ui，，1 个测试。

项目亮点/ 难点：

在做题的时候，会有一个书写功能，点击之后会有一个遮罩层，可以让用户

在屏幕上打草稿，比较方便。这个需求还是比较少见的，有点类似于签名功能，实现方法也不止一种，用 SVG 应该也可以，但是我们是基于 canvas 元素封装了一个组件来实现。实现的原理就是先准备一个绘图的环境，然后配置一下线条的宽度，样式，然后通过监听画布事件来实现连贯执行绘制，核心的内容就是计算、描绘划过的路径。Canvas 画布为我们提供了一个用来作图的平面空间，该空间的每个点都有自己的坐标。然后我们通过一个（getBoundingClientRect）方法获得页面 Canvas元素相对视窗的位置左边和顶部的像素距离，再利用 clientX，clientY 事件属性返回当事件被触发时鼠标指针向对于页面的水平和垂直坐标，最后通过 lineTo和 stroke 来绘制路径。在 events 事件执行之后要移除，否则会造成一直绘制。然后在画布右上角还有一个清除功能，能够清除已经绘制的路径，是用 Canvas 的 clearRect 方法就可以实现清除画布区域内容。