项目功能点

一、登录功能

1.1 登录实现思路

1. 点击登录按钮获取到输入框输入的账号与密码
2. 调用登录接口将获取到的用户名和密码发送给后台
3. 登录成功之后后台会返回用户信息以及token
4. 将token和用户信息保存到vuex和本地
5. 跳转到主页面

1.2 面试官可能会问到的问题

1.2.1 什么是token？ token是怎么来的?

- 所谓的token其实就是一段加密的信息
- token是后台通过**json web token**(简称为:jwt)来生成的,这个签证信息由三部分组成,分别
- 是:header 、 payload、secret.token是后台生成的,前端登录成功之后后台会将生成的token发送给前端

1.2.2 如何将token发送给后台?

- 在axios的请求拦截器中发送,代码为: config.header.Authorization = token

1.2.3 token过期如何处理?

- token的过期时间是由后端来决定的
- 前端如果想知道token有没有过期,需要根据后端返回的状态码来判断,如果状态码为401则表示token过期
- token过期之后我们需要调用退出登录方法,并且将本地和vuex里面存储的token进行清除

二、页面鉴权

> 所谓的页面鉴权其实指的是当用户在未登录状态时,无法访问非登录页面、404、401以外的页面,当用户为登录状态时,可以访问登录以外的页面

2.1 页面鉴权的实现思路

当用户登录成功时,我们会将后台返回的token保存到vuex和本地,并且跳转到主页,这时我们会触发vue-router给我们提供的路由守卫
方法,在路由守卫方法里面我们会先获取vuex或者本地存储的token,如果token存在的话则通过next进入主页,如果token不存在的话,则通过next方法跳转到登录页面

2.2 对应代码

```
/**
 *
 * 有token
 *    可以进入非登录页面
 * 、  不能在进入登录页面
 * 没有token
 *    只能进入登录页面或者401/404
 * **/
```

```
9
10  import router from "./router/index.js"
11  import store from "./store/index.js"
12  // 白名单
13  const whiteList = ['/login',"/404","/401"]
14
15  router.beforeEach((to, from, next) => {
16
17    if (store.getters.token)
18      { if (to.path === '/login')
19      {
20        next('/')
21      } else {
22        next()
23      }
24    } else {
25      if(whiteList.indexOf(to.path) > -
26        1){next()
27      }else{
28        next("/login")
29      }
30
31  })
32
```

三、RBAC权限

> 所谓的RBAC（Role-Based Access Control，基于角色的访问控制），其实就是用户通过角色与权限进行关联。简单地说，一个用户拥有若干角色，每一个角色拥有若干权限。这样，就构造成“用户-角色-权限”的授权模型。在这种模型中，用户与角色之间，角色与权限之间，一般者是多对多的关系

在实际项目开发过程中,我们会涉及到以下的一些权限:

3.1 菜单权限

所谓的菜单权限其实指的是用户后台主页的菜单栏的数据是动态的,而不是写死的,如果要想把菜单栏的数据变成动态的,我们一般是通过两种方案来解决:

这块大家详细说第一种方案就行,第二种方案可以稍微提一下

1. 第一种方案
    - **实现思路**

      菜单的数据一般是当我们登录成功之后,后台会给我们返回token以及用户相关信息,其中用户信息这块会包含菜单的数据、路由的数据、按钮权限的数据. 我们需要将这些数据存储到vuex和本地.

      接下来我们需要在主页组件中获取到vuex里面所存储到的菜单数据,利用 `el-menu` 将菜单的数据进行动态渲染.
    - **难点:**

      但是菜单渲染这块有一个难点,后台给我们返回的菜单数据一般会是一个tree结构的数据,但是我

们无法预测这个tree结构的数据有多少层,如何直接使用el-menu进行渲染的时候就会存在一个问题,无法将tree结构的数据完全渲染出来,所以这块我们需要使用vue递归组件去实现

- **面试官可能问到的问题:**

- 什么是递归?

  所谓的递归,实际指的是函数调用函数自身,但符合结束条件时,函数则结束自身的调用,如果不设置结束条件,则会造成死循环.

- 什么是递归组件?

  所谓的递归组件就是自己的内部实现又调用自己的组件,我们称之为递归组件,使用递归组件时我们需要注意两点,
  一是需要给组件提供name选项,二是要设置终止条件,否则会无限递归,造成调用栈溢出

- 你这个递归组件是如何实现的?

  1. 在components目录下创建一个TreeMenu组件
  2. 使用template标签包裹el-submenu组件
  3. 在template组件上使用v-for渲染菜单的数据
  4. 在el-submenu设置终止条件,如果menu.children是否存在以及menu.children.length的长度是否大于0,如果不存在或者长度不大于0,则终止递归组件的调用
  5. 具体的代码

```
<template>
  <template v-for="menu in userMenu">
    <el-submenu
      v-if="
        menu.children &&
        menu.children.length > 0 &&
        menu.children[0].menuType == 1
      "
      :key="menu._id"
      :index="menu.path"
    >
      <template #title>
        <i :class="menu.icon"></i>
        <span>{{ menu.menuName }}</span>
      </template>
      <tree-menu :userMenu="menu.children" />
    </el-submenu>
    <el-menu-item
      v-else-if="menu.menuType == 1"
      :index="menu.path"
      :key="menu._id"
      >{{ menu.menuName }}</el-menu-item
    >
  </template>
</template>
<script>
export default {
  name: "TreeMenu",
  props: {
    userMenu: {
      type: Array,
      default() {
        return [];
      },
    },
  },
};
</script>
```

- 你是怎么做到不同的用户登录显示不同的菜单?

  这个其实很简单,我们菜单的数据是后台根据不同的用户登录给前台返回不同的菜单数据,所谓

前台只需要做到将菜单的数据渲染出来就行,至于这个用户拥有哪些菜单数据完全是由后台来决定的

2. 第二种方案

- **实现思路**

  菜单的数据不是后端来返回,而是前端通过动态路由表来生成菜单的数据. 具体实现思路是通过this.$router.options.routes获取到完整的路由表,然后在封装一个递归方法,过滤出菜单所需要的数据,最后在通过el-menu渲染成菜单.

3.2 路由权限

> 所谓的路由权限指的是项目中的路由数据不是写死的,而是通过后台返回的路由权限数据进行动态渲染的. 如果用户在浏览器地址访问了不存在的路由,我们则跳转到404页面

- **实现思路**
  1. 当用户登录的时候,调用getPermissionList接口获取到后台所返回的路由表数据
  2. 定义一个递归方法,将后台所返回的路由表数据过滤成路由所需要的数据
  3. 在通过router.addRoutes动态的添加到路由表当中

3.3 按钮权限

- **什么是按钮权限?**

  所谓的按钮权限是指在某个菜单的界面中， 我们需要根据按钮权限数据， 展示出可进行操作的按钮，比如删除， 修改， 增加等按钮.

- **如何实现按钮的权限控制?**

  如果要实现按钮的权限控制,我们需要使用vue的自定义指令去实现:

```
1    1. 首先需要创建一个按钮权限控制的指令,我们定义这个指令的名称为: `v-permission`
```

  2. 在这个指令的内部获取到vuex里面存储的按钮权限数据
  3. 在通过binding.value获取到自定义制定属性值的数据
  4. 判断从vuex里面获取到的按钮权限数据是否包含了自定义指令包含的权限
  5. 如果不包含,我们在设置el.style.display = “none”,并且使用el.parentNode.removeChild(el)删除当前按钮元素

- **具体实现代码:**

```
1   Vue.directive('permission', {
2       beforeMount: function (el, binding) {
3           let actionList = storage.getItem('actionList');
4           let value = binding.value;
5           let hasPermission = actionList.includes(value)
6           if (!hasPermission) {
7               el.style = 'display:none';
8               setTimeout(() => {
9               el.parentNode.removeChild(el);10
                }, 0)
11          }
12      }
13  })
```

3.4 接口权限

> 所谓的接口权限其实指的是当用户在未登录时无法请求到登录以后才当请求的接口, 如果为登录时直接请求,后台会返回401.
>
> 如果要实现接口权限校验,我们需要登录以后将后台所返回的token保存到vuex和本地, 在axios请求拦截器中使用config.header.token = token 或者config.header.Authorization = token 的方式将token使用请求头发送给后台

四、其他功能点大家可自行补充

4.1 功能点1

4.2 功能点2

4.3 功能点3

4.4 功能点4

面试录音题

- 宋星凯的所有面试录音问题
  待整理
- 任振旭的所有面试录音问题
  待整理
- 宫玉涛的所有面试录音问题
  待整理
- 陶家庆的所有面试录音问题
  待整理
- 孙嘉梁的所有面试录音问题
  待整理
- 张梅荣的所有面试录音问题
  待整理
- 郭子尧的所有面试录音问题
  待整理
- 张梦涵的所有面试录音问题
  待整理
- 白文山的所有面试录音问题
  待整理
- 袁舒文的所有面试录音问题
  待整理
- 任宏日的所有面试录音问题
  待整理
- 周海超的所有面试录音问题
  待整理
- 张嘉鑫的所有面试录音问题
  待整理
- 沙小双的所有面试录音问题
  待整理
- 王旭的所有面试录音问题
  待整理
- 敖日格勒的所有面试录音问题

待整理

- 安栋锆的所有面试录音问题
  待整理
- 蔡福晴的所有面试录音问题
  待整理
- 刘旭照的所有面试录音问题
  待整理
- 李佳升的所有面试录音问题
  待整理
- 丁嘉伟的所有面试录音问题
  待整理
- 董靖楠的所有面试录音问题
  待整理
- 何新统的所有面试录音问题
  待整理
- 龙智宇的所有面试录音问题
  待整理
- 程旭东的所有面试录音问题
  待整理
- 储宇航的所有面试录音问题
  待整理

面试题复习

面试题复习

一、vue面试题

1. vue生命周期? (必问)
    1.1 什么是vue生命周期?
    1.2 vue生命周期都有哪些钩子函数?这些钩子函数如何触发?
    1.3 项目开发过程中,在生命周期里面都分别做过什么功能?
2. vuex的理解? (必问) 讲解
    2.1 什么是vuex?使用vuex能够解决什么问题?
    2.2 vuex的五大核心是什么?
    2.3 在组件里面如何调用五大核心的属性和方法?
    2.4 vuex的执行机制是什么?
    2.5 vuex的弊端是什么?怎么解决?
3. vue路由有几种模式?有什么区别?原理是什么?(必问) 讲解
4. vue路由守卫?(必问) 讲解页面权限
5. v-if与v-show的区别?(90%)
6. v-for与v-if的优先级那个高?如果同时使用v-for和v-if怎么解决?(90%) 讲解
7. methods、computed和watch的区别?(90%)。讲解
8. vue组件通信?(必问)。eventBus 讲解
9. $nextTick方法有什么作用?(80%) 讲解
10. 说一下什么是mvvm模式?(70%)
11. vue双向数据绑定原理?(必问) 过一下
12. vue常用的指令有哪些?(50%)
13. vue常用的修饰符有哪些?(50%)
14. vue如何封装可复用的组件?以及请举例说明你封装过的组件?(50%) 讲解
15. vue中key的作用是什么?(必问)。过一下
16. 说一下你对keep-alive的理解?以及在项目中如何使用?(90%) 过一下
17. 说一下什么是vue过滤器? 有几种?项目中如何使用,请举例说明?(60%)
18. 说一下你对slot插槽的理解?(70%) 过一下
19. 说一下vue中本地跨域如何解决?线上跨域如何解决?(必问) 线上在讲解一下
20. 说一下如何对axios进行二次封装?以及api如何封装?(30%)
21. 说一下axios的拦截器的作用?应用场景都有哪些?(80%)
22. 说一下vue和jquery的区别?(50%)
23. vue中data发生变化,视图不更新如何解决?(必问) 过一下
24. 为什么vue中data必须是一个函数?(必问) 过一下

二、ES6面试题

1. 说一下你对promise的理解?(必问)
    1.1 什么是promise?通过promise能够解决什么问题?
    1.2 说一下promise的特点?也就是三种状态?
    1.3 说一下promise怎么用?
    1.4 在说一下promise的all方法和race方法?
    1.5 在说一下在项目中使用promise做过什么?
2. 说一下async和await、以及他们和promise的区别?(必问)
3. 说一下es6新增的特性有那些?(必问)
4. 说一下数组新增的方法有哪些?这些方法分别是什么意思?(必问)
5. 说一下map方法、forEach方法、filter方法的作用以及他们之间的区别?(90%)
6. 说一下var 、let、const之间的

区别?(95%)7.说一下箭头函数与

普通函数的区别?(80%)

8. 说一下for in 与for of的区别?(80%)
9. 说一下es6如何实现类以及如何实现类的继承?(80%)
10. 说一下数组去重的方法有哪些?es6如何实现数组去重?(90%)
11. 说一下如何检测对象里面有没有属性(或者如何检测一个对象是否为空)?以及如何获取对象里面所有的属性名?(70%)
12. 如何将多个数组合并成为一个数组?(70%)
13. 说一下forEach、map、filter、reduce、some、every

等方法的作用?(50%) 三、JavaScript面试题

1. 什么是浅拷贝？如何实现浅拷贝？什么是深拷贝？如何实现深拷贝？(必问)
2. 什么是闭包?(必问)
3. 说一下什么是原型?什么是原型链?(90%)

4. 实现数组去重?最少写出三种方法?(70%)

一、vue面试题

1. vue生命周期? (必问)

1.1 什么是vue生命周期?

1.2 vue生命周期都有哪些钩子函数?这些钩子函数如何触发?

1.3 项目开发过程中,在生命周期里面都分别做过什么功能?

> 面试官您好,我先介绍一下什么是vue的生命周期? 所谓的vue生命周期就是vue实例从创建到销毁的整个过程我们称之为vue的生命周期,通过vue的生命周期我们可以在不同的阶段进行不同的逻辑操作. vue生命周期常用的钩子函数一共有8个,分别是创建前后、挂载前后、更新前后以及销毁前后. 分别对应的钩子函数为beforeCreate 创建前、 created创建后、beforeMount 挂载前、mounted挂载后、beforeUpdate 更新前、updated更新后、beforeDestory 销毁前、 destoryed销毁后, 页面一开始加载的时候就会触发创建前后和挂载前后的钩子函数, 而更新的钩子函数需要当我们改变data的时候才能触发,比如 点击按钮,执行一个方法,在这个方式里面给data里面属性重新进行复制操作,这个时候就会更新的钩子函数, 销毁的钩子函数必须得当组件进行切换的时候就会进行销毁.
>
> 在项目开发过程中,我经常使用到的钩子函数有created,我们经常在created进行数据请求,或者获取本地存储的数据,还有一些其他的操作. 除了created还有mounted,我们经常在mounted里面获取dom元素 (有时 候也存在获取不到dom元素的情况,这个时候我们一般用$nextTick方法来解决). 以上就是我对生命周期的理解

2. vuex的理解? (必问) 讲解

2.1 什么是vuex?使用vuex能够解决什么问题?

2.2 vuex的五大核心是什么?

2.3 在组件里面如何调用五大核心的属性和方法?

2.4 vuex的执行机制是什么?

2.5 vuex的弊端是什么?怎么解决?

面试官您好,接下来我先给您介绍一下什么是vuex? 所谓的vuex其实就是vue官方给我们提供的一个状态管理工具,通过vuex我们可以组件之间数据共享的问题. vuex一共有5大核心,分别是state,里面保存的是状态,也可以理解为是数组, 接下来是getters, 他们用来获取state里面的状态,并且可以对state的数据进行处理之后在返回,有点类似于vue的计算属性, 接下来还有mutations,他的作用主要是用来修改state里面的数据,不过在mutations里面只能进行同步的操作,还有actions,这个actions也可以去改变state的状态,不过在actions里面定义的方法可以进行异步操作,最后一个是modules,如果当我们的项目比较大的时候,那么保存的状态也会增加,如果都写到index.js文件里面,文件的内容就会变得特别臃肿,后期难以维护,所以我们可是使用modules进行模块化的处理,将多个状态抽离到对应js文件里面,最后在modules进行合并,这样后期就方便维护了.

接下来我在介绍一下在组件里面如何调用vuex里面的属性和方法,如果我们要获取state里面的状态,我们可以通过this.

store.commit来触发,如果调用actions里面的方法,我们一般通过this.$store.dispacth来进行触发. 除了这种方式以外,我们还可以通过辅助函数的方式来进行调用和触发(mapState, mapMutation,mapAction, mapGetter)

我在项目当中如果要改变state的状态,我们一般是在组件里面调用this.$store.dispatch方式来触发actions里面的方法,在actions
里面的方法通过commit来调用mutations里面定义的方法来改变state,同时这也是vuex的执行机制

不过vuex也有一些弊端,比如浏览器刷新的时候,vuex的数据会丢失,我们一般结合本地存储来解决,当我们在mutations里面改变state的时候在通过localStorage或者sessionStorage存储到本地,然后在state的状态的属性值那块写一个三元表达式,如果本地存在数据的话则读取本地存储的数据,否则就赋值为null

在项目当中我一般使用vuex会保存用户信息和token以及其他的一些状态. 以上就是我对vuex的理解.

3. vue路由有几种模式?有什么区别?原理是什么?(必问) 讲解

面试官您好,接下来我给您介绍一下vue的路由模式,vue的路由模式一共有两种,分别是哈希和history. 他们的区别是hash模式不会包含在http请求当中,并且hash不会重新加载页面,而使用history模式的话,如果前端的url和后端发起请求的url不一致的话,会报404错误,所以使用history模块的话我们需要和后端进行配合.

history的原理就是利用html5新增的两个特性方法,分别是psuhState和replaceState来完成的. 以上就是我对vue路由模式的理解.

4. vue路由守卫?(必问) 讲解页面权限

面试官您好,接下来我给您介绍一下vue路由守卫,首先呢,所谓的路由守卫就是当我们进行页面跳转的时候会触发的钩子函数,我们把它称之为vue路由守卫. vue一共给我们提供了三种路由守卫,第一种全局路由守卫,第二种是组件内路由守卫,第三种路由独享守卫,这个是写在路由里面. 不管是全局,还是组件以及独享,都会有beforeEach、beforeResolve 、afterEach 分别表示的意思路由跳转前会触发的钩子函数以及进入路由的时候,以及进入路由之后会触发的钩子函数. 这几个钩子函数里面都有一个回调函数,这个回调函数里面会有三个参数,分别是to,from,next,分别对应的是要进入的路由、离开之前的路由,以及进入写一个路由

在项目中我们经常使用路由守卫实现页面的鉴权. 比如:当用户登录之后,我们会把后台返回的token以及用户信息保存到vuex和本地,当页面进行跳转的时候,我们会在路由守卫里面获取vuex里面的token,如果token存在的话,我们则使用next让他进入要跳转的页面,如果token不存在的话我们使用next方法让他回到登录页

以上就是我对vue路由守卫的理解

5. v-if与v-show的区别?(90%)

面试官您好,接下来我给您介绍一下v-if和v-show的区别? 首先v-if和v-show都是控制元素的显示与隐藏, 不过v-if控制元素的显示和隐藏的时候会删除对用的dom元素,当每一个显示的时候,都会重新创建dom和渲染. 而v-show则是通过css的display:none和display:block来控制元素的显示与隐藏. v-if比较耗费性能,所以我们涉及到频繁的显示隐藏操作我们建议使用v-show,如果不是频繁操作的话,我们可以v-if

在项目中我会经常使用v-if和v-show,比如我们在搜索功能的时候,他有一个历史记录,这个时候我们根据是否有搜索的结果来判断历史记录的显示与隐藏,这块我就可以使用v-if ,当然用v-show也可以. 以上就是我对v-if和v-show的理解.

6. v-for与v-if的优先级那个高?如果同时使用v-for和v-if怎么解决?(90%) 讲解

v-for的优先级高. 因为v-for的时候我们才开始渲染dom元素,这个v-if还无法进行判断.

v-for和v-if不能同时使用,我们可以通过标签,比如div或者template标签来进行包裹,把v-if写到包裹的标签上面(写到v-for外面)

7. methods、computed和watch的区别?(90%)。讲解

首先呢,methods是用来定义方法的区域,methods定义的方法需要调用才能触发. 不具备缓存行

而computed是计算属性,他依赖于属性值的变化,当属性发生改变的时候,计算属性里面定义的方法就会触发,computed具有缓存性,依赖属性值的变化而变化.

而watch主要是用于监听,不具备被缓存性.依赖于数据变化而触发.

在项目中,比如我们获取state的状态的时候我们会把它放到computed里面,或者在写购物车数量计算的时候也会使用计算属性.

而watch也在项目经常使用,比如我们封装编辑 和 新增弹窗组件的时候会通过watch来进行id判断我们要显否要清空表单的数据.

以上就是我对computed和watch的理解.

8. vue组件通信?(必问)。eventBus 讲解

父传子 在子组件的标签上定义属性 子组件通过props来进行接受,可以通过数组的方式进行接受,也可以通过对象的方式来进行接收,如果父组件没有传递属性,子组件可以default来设置默认值

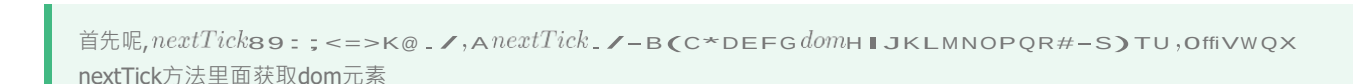
子传父 子组件通过this.$emit("自定义的事件",要传给父组件的数据), 父组件通过子组件的标签监听自定义的时间,通过方法来接收传递的数据

非父子组件通信

通过中央事件总线,我们也称之为eventBus,

我们需要创建一个空的js文件,在这个文件里面创建空的vue实例,然后导出这个空的vue实例,通过实例对象调用.on方法进行接收,通过emit方法来进行发送,以上就是非父子组件通信的方式

9. $nextTick方法有什么作用?(80%) 讲解

首先呢, 
nextTick方法里面获取dom元素

10. 说一下什么是mvvm模式?(70%)

> MVVM 是把 MVC 的 Controller 和 MVP 的 Presenter 改成了 ViewModel 。
>
> View 的变化会自动更新到 ViewModel · ViewModel 的变化也会自动同步到 View 上显示。这种自动

同步是因为 ViewModel 中的属性实现了 Observer ，当属性变更时都能触发对应的操作。

11. vue双向数据绑定原理?(必问) 过一下

vue.js 则是采用 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。

这个时候就可以实现数据的双向绑定

12. vue常用的指令有哪些?(50%)

v-if

v-show

v-html

v-text

v-on

v-bind

v-model

v-for

13. vue常用的修饰符有哪些?(50%)

.trim 去除首尾多余的空格

.stop 阻止事件冒泡

.once 只渲染一次

.self 事件只作用在元素本身

.number 将值转化为number类型

.capter 组件之间捕获

.prevent 阻止元素的默认行为

.native 事件穿透,让我们可以在自定义组件上定义事件和方法

14. vue如何封装可复用的组件?以及请举例说明你封装过的组件?(50%) 讲解

1. 分析项目的所有页面结构和业务功能,抽离出相同的页面结构和业务功能
2. 在src目录下创建一个components这个的文件夹
3. 在这个文件夹内创建可复用的组件
4. 在需要的用的组件里面引入创建的这个可复用的组件,并进行注册,以标签的形式写在对应的地方
5. 接下来就需要对可复用的组件内容要进行封装,那么在封装的时候我们要注意组件的封闭性和开放性以及粗细粒度
6. 所谓的这个封闭性就是当我们在组件内部定义好之后外部是无法进行修改的,比如当前有一个关闭的符号,或者有一个箭头,我们需要不管任何人使用该组件时候都能够显示这个箭头,无法从外部进行修改
7. 所谓的开放性就是我们需要将动态的内容以及组件通信的方式进行数据传递,保证组件的可扩展性

8. 而所谓的粗细力度就是我们可以把一整个页面当作一个组件去进行封装,也可以一个页面包含了多个组件,至于到底选择哪种呢,这个是没有一个标准的,我们需要根据自己的业务需求去进行判断
9. 总结来说,所谓的如何封装可复用组件其实技术核心就是通过我们vue提供的组件通信在结合slot插槽来进行分装
10. 比如:封装一个搜索框组件:
    1. 在components里面创建search.vue
    2. 在search.vue里面实现搜索框的布局
    3. 在props里面接受 title, bgColor, size , icon,以及当点击搜索按钮或者点击回车键的时候,触发一个方法,通过this.$emit将输入框输入的值传递给父组件
    4. 接下来要使用这个搜索组件,我们需要通过import 在父组件内引入子组件,并在componetns属性里面进行注册,
    5. 在页面就可以使用,这个时候我们可以通过传递titile老控制子组件搜索框显示的内容,通过bgcolor可以控制搜索框的背景颜色,也可以通过size设置搜索框字体的大小,也可以通过icon来设置搜索框的图标, 通过监听$emit里面定义的方法来获取搜索框输入的值

    以上就是封装的过程

15. vue中key的作用是什么?(必问)。过一下

避免dom元素重复渲染. 我门一般在设置key的时候首先尽量会设置为id,或者index下表.

16. 说一下你对keep-alive的理解?以及在项目中如何使用?(90%) 过一下

keep-alive是vue内置的一个组件，而这个组件的作用就是能够缓存不活动的组件，我们能够知道，一般情况下，组件进行切换的时候，默认会进行销毁，如果有需求，某个组件切换后不进行销毁，而是保存之前的状态，比如说刚刚填好的表单数据。那么就可以利用keep-alive来实现

在搭建 vue 项目时，有某些路由组件没必要多次渲染，所以需要将组件在内存中进行'持久化'，此时在router-view上使用keep-alive。 keep-alive可以使被包含的路由组件状态维持不变，即便是组件切换了，其内的状态依旧维持在内存之中。在下一次显示时，也不会重新渲染。

include - 字符串或正则表达式。只有名称匹配的组件会被缓存。 exclude - 字符串或正则表达式。任何名称匹配的组件都不会被缓存。 max-数字最多可以缓存多少组件。

以上就是我对keep-alive的理解

17. 说一下什么是vue过滤器? 有几种?项目中如何使用,请举例说明?(60%)

所谓的vue过滤器就是将数据进行二次处理,得到我们想要的结果数据

vue的过滤器分为两种,第一种是全局过滤器,通过vue.filet来进行定义,第二种是局部过滤器,需要定义在组件内部

项目中我们通过过滤器将后台返回的状态0 和1 转化为支付或者未支付

18. 说一下你对slot插槽的理解?(70%) 过一下

首先呢,所谓的插槽就是一个占位符,将自定义组件的内容展示出来.我们知道自定义的组件里面如果写内容的话,页面是不会显示出来的,如果我们想让自定义组件里面的内容显示出来,我们就需要使用slot的插槽.

而插槽分别具名插槽和匿名插槽、以及作用域插槽. 我们用的比较多的具名插槽和匿名插槽,具名插槽需要所有slot标签上指定name属性,而在对应标签上添加v-slot属性.

在项目中我们一般在进行组件封装的时候会使用插槽,以上就是我对插槽的理解.

19. 说一下vue中本地跨域如何解决?线上跨域如何解决?(必问)　线上在讲解一下

本地跨域是通过在vue.config.js文件里面的devServer属性里面的proxy属性里面配置,一共配置三个属性,分别是代理名称 代理地址 开启跨域 重写路径

线上跨域是在nginx.conf文件里面配置, 代理名称是通过location 代理名称。proxy_pass 代理地址

20. 说一下如何对axios进行二次封装?以及api如何封装?(30%)

1. 在src文件夹内创建utils文件夹
2. 在utils文件夹内创建request.js文件
3. 在request.js内引入axios
4. 使用axios.create方法创建axios的实例,在axios.create方法里面可以配置请求的公共地址和超时时间以及其他的一些配置
5. 在创建请求拦截器和响应拦截器
6. 在请求拦截器里面可以获取vuex的token,并通过config.header.token　=　vuex的token,将token发送给后台
7. 在请求拦截器里面我们配置loading加载
8. 在响应拦截器里面我们可以结束loading加载以及token的过期处理,以及错误响应信息的处理
9. 最后通过export default 导出axios的实例对象
10. 在src文件内创建api文件夹
11. 在api文件夹内创建对应模块的js文件
12. 在对应的文件里面引入request.js文件
13. 封装api方法
14. 最后通过export default 导出封装的api方法

21. 说一下axios的拦截器的作用?应用场景都有哪些?(80%)

首先呢,axios拦截器是axios给我们提供的两个方法,通过这两个方法我们可以对请求发送之前以及响应之后进行逻辑的再次处理(拦截). 这两个拦截器不需要手动触发,只要发送http请求的时候就会自动触发. 我在项目中经常通过拦截器发送token, 对token进行过期处理,以及进行其他的一些操作

22. 说一下vue和jquery的区别?(50%)

首先呢jquery他是用js封装的一个类库,主要是为了方便操作dom元素,而vue他是一个框架,并且呢,他会从真实dom构建出一个虚拟的dom树,通过di"算法渲染只发生改变的dom元素,其他的相同的dom元素不用在重新渲染.　而使用jquery去改变dom元素的时候,即使有相同的dom元素也会重新渲染,以上就是我对vue和jquery区别的理解.

23. vue中data发生变化,视图不更新如何解决?(必问) 过一下

面试官,您好,接下来我先给您介绍一下为什么data发生变化,视图不更新,**因为Vue实例中的数据是响应式的****而我们新增的属性并不是响应式的．由于受现在JavaScript的限制．Vue无法检测到属性的新增或删除。所以有时无法实时的更新到视图上。**

所以我在项目中遇到这类问题的时候一般是通过this.$set - ZYZ.this.$set方法一共有三个参数,分别是目前属性,新增属性,新增的值.

以上就是我对视图不更新的理解.

24. 为什么vue中data必须是一个函数?(必问) 过一下

> 如果data是一个函数的话，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。
>
> 所以说vue组件的data必须是函数。这都是因为js的特性带来的，跟vue本身设计无关。

二、ES6面试题

1. 说一下你对promise的理解?(必问)

1.1 什么是promise?通过promise能够解决什么问题?

1.2 说一下promise的特点?也就是三种状态?

1.3 说一下promise怎么用?

1.4 在说一下promise的all方法和race方法?

1.5 在说一下在项目中使用promise做过什么?

> 首先promise是es6提供一种异步解决方案. 通过promise能够解决回调地狱问题. 所谓的这个回调地狱指的当我们执行完一个操作之后在接受着操作的结果只能通过回调函数的方式进行接受,使用回调函数的方式存在的弊端就是写法非常臃肿,并且后期难以维护,所谓es6给我提供了一种新的解决方案,就是promise来进行解决,promise可以通过链式调用的方式来解决层层嵌套的问题,但是写法上也不是非常好,所以我们最终的替代方案是使用async和await
>
> promise一共有三个状态,分别是进行中,成功或者失败 如何成功的话可以通过resolve方法将正确结果返回出去,通过.then的方式进行接受,失败的话可以通过reject的方式将失败的结果返回出去,通过.catch的方式进行接受,pending状态是进行中,一旦进行之后,他的状态是不可逆的
>
> 如果要使用promise,我们需要对promise进行实例化,在实例化的构造函数里面有一个回调函数,这个回调函数里面有那个参数,分别是resolve和reject,我们可以通过promise的实例化对象调用.then或者.catch方式接受结果
>
> promise还给我们提供了.all 和 race, 其中all方法的作用是将多个请求合并成一个请求, 比如当首页要请求10个接口,我们可以promise.all进行合并,.race的作用也可以将多个请求合并成一个请求,不过是谁先请求成功就先返回谁.
>
> 我在项目中经常使用promise对api 截形封装以及如何a页面要获取到b页面的结果,我们也可以结果promise来完成
>
> 以上就是我对promise 的理解.

2. 说一下async和await、以及他们和promise的区别?(必问)

> 首先async和await是解决异步的终极方案,他是generatal的语法糖. async和await一般配和使用,当我们给函数前面加上关键字async,这个时候,这个函数的返回值就是一个promise. 而await是一个同步的操作,await只能配合async只能,不然会报错,await后面可以是表达式,也可以是一个promise,在await下面的代码必须得等待await执行完之后才能在执行
>
> 他们和promise的区别就是在写法上更加的简洁.
>
> 以上就是我对async和await的理解.

3. 说一下es6新增的特性有那些?(必问)

> 1. 新增了变量声明方式,也就是let和const
> 2. 新增了解构赋值
> 3. 新增了一个数组方法 字符串方法 正则表达的方法 函数的一些写法 对象的方法
> 4. promise

5. async await
6. class 以及 继承
7. 模块化
8. 新的数据类型
9. 大概能想到暂时只有这么多,在项目中我经常使用let和const 箭头函数。解构赋值 promise 还有 async await

4. 说一下数组新增的方法有哪些?这些方法分别是什么意思?(必问)

Array.from方法 将类数组转化为真正的数组

Array.of方法 将数值转化为数组

copyWithin() 方法是将指定位置的成员复制到其他位置（会覆盖原有成员）

find方法和findindex方法 查找符合条件的元素。 查找符合条件元素的下表

includes方法可以查看表个数组是否包含给定的值

flat方法将多维数组转化为1维数组或者指定维度的数组

遍历的数组的方法:

forEach 类似于for循环,主要是用来遍历数组

map方法 主要作用是映射一个新的数组,可以对数组进行遍历

filter方法 主要作用返回一个符合条件新的数据,同样也有遍历数组的作用

5. 说一下map方法、forEach方法、filter方法的作用以及他们之间的区别?(90%)

forEach() 方法: 循环原来的数组

map() 方法： 循环原数组并映射一个新数组出来

filter() 方法： 过滤不需要的数组元素

6. 说一下var 、let、const之间的区别?(95%)

- `var` 存在提升，我们能在声明之前使用。 `let` 、 `const` 因为暂时性死区的原因，不能在声明前使用
- `var` 在全局作用域下声明变量会导致变量挂载在 `window` 上，其他两者不会
- `let` 和 `const` 作用基本一致，但是后者声明的变量不能再次赋值。

7.说一下箭头函数与普通函数的区别?(80%)

在es6中，提供了一种简洁的函数写法，我们称作"箭头函数"。

写法：函数名=(形参)=>{.....}　当函数体中只有一个表达式时，{}和return可以省略，当函数体中形参只有一个时，()可以省略。

特点：箭头函数中的this始终指向箭头函数定义时的离this最近的一个函数，如果没有最近的函数就指向window。

区别:

1. 箭头函数不能用于构造函数，不能使用new** 而普通函数可以
2. 在普通函数中，this总是指向调用它的对象，如果用作构造函数，this指向创建的对象实例,而箭头函数指向箭头函数定义时的离this最近的一个函数，如果没有最近的函数就指向window。

8. 说一下for in 与for of的区别?(80%)

For in可以遍历对象 而 for of遍历对象会报错

for in 遍历数组得到的数组的下表 而for of遍历得到的时候数组里面的每一个元素

9.说一下es6如何实现类以及如何实现类的继承?(80%)

es6提供了类的这个概念,在es5中是没有类的这个概念,如果想在es5中实现一个类型,我们只能构造函数的方式去创建一个类,而es6给我们提供一个更方便 的方.法,那就是class,这个class理解为是构造函数的语法糖.

我们创建一个类只需要用过关键词class去声明就可以了，他的调用方式和构造函数的调用方式是一样的

通过es6的类还给我们提供一个extends这样的一个关键字,来实现继承

以上就是我对类的理解

10. 说一下数组去重的方法有哪些?es6如何实现数组去重?(90%)

indexof

双层for循环

set方法

11. 说一下如何检测对象里面有没有属性(或者如何检测一个对象是否为空)?以及如何获取对象里面所有的属性名?(70%)

通过object.keys方法, 返回值数组,数组里面包含的是所有属性名

Object.hasOwnProperty()

使用for in的方式

12. 如何将多个数组合并成为一个数组?(70%)

es5 :

concat

for循环

Es6:

扩展运算符

map方法

13. 说一下forEach、map、filter、reduce、some、every等方法的作用?(50%)

reduce 遍历数据求和。

some 遍历数组每一项，有一项返回true,则停止遍历，结果返回true。不改变原数组

遍历数组每一项，每一项返回true,则最终结果为true。当任何一项返回false时，停止遍历，返回false。不改变原数组

forEach() 方法: 循环原来的数组

map() 方法： 循环原数组并映射一个新数组出来

filter() 方法： 过滤不需要的数组元素

三、JavaScript面试题

---

1. 什么是浅拷贝？如何实现浅拷贝？什么是深拷贝？如何实现深拷贝？(必问)

浅拷贝只复制指向某个对象的指针，而不复制对象本身，新旧对象还是共享同一块内存。浅拷贝只复制

对象的第一层属性

但**深拷贝**会另外创造一个一模一样的对象，新对象跟原对象不共享内存，修改新对象不会改到原对象。

对对象的属性进行递归复制

浅拷贝的方式:

使用Object.assign({},obj)第一个参数是一个空对象，第二个参数是你要复制的对象；通过这

个方法我们知道浅拷贝不能修改基础的数据类型，可以修改引用的数据类型；

ES6中的...扩展运算符来进行浅拷贝的实现；

Object.assign()实现

深拷贝的方式:

对象只有一层的话可以使用上面的：**Object.assign()　函数**

用JSON.stringify把对象转成字符串，再用JSON.parse把字符串转成新的对象。

2. 什么是闭包?(必问)

说白了就是函数嵌套函数，内部函数能够访问外部函数的变量

闭包的优点

可以隔离作用域，不造成全局污染

闭包的缺点

由于闭包长期驻留内存，则长期这样会导致内存泄露

如何解决内存泄露：将暴露全外部的闭包变量置为null

适用场景：封装组件，for循环和定时器结合使用,for循环和dom事件结合.可以在性能优化的过程

中,节流防抖函数的使用,导航栏获取下标的使用

3.说一下什么是原型?什么是原型链?(90%)

当对函数进行实例化的时候就会产生一个prototype,而这个prototttype就是原型属性,proto才是真正的原型,通过这个**原型**属性我们就可以访问另一个对象所有的属性和方法

当我们将多个实例挂载对应的原型上的时候,这个时候要一层一层往上的查找,这个时候就会形成原型链

4. 实现数组去重?最少写出三种方法?(70%)

indexof

var arr = [1,1,1,1,2,3,4,5,6];

var temp= [];

for(var

i=0;i<arr.length;i++){ if(temp.i

ndexof(arr[i] == -

1)){temp.push(arr[i])s

}

}

set

new Set(arr)

双层for循

5. 什么是递归?如何用代码实现递归?(60%)

函数本身调用自身,满足对应的条件则进行停止

6. 请说出实现冒泡排序的思路?(40%)

1. 比较相邻的两个元素，如果前一个比后一个大，则交换位置。 2、比较完第一轮的时候，最后一个元素是最大的元素。 3、这时候最后一个元素是最大的，所以最后一个元素就不需要参与比较大小。
2.
3. 双层for循环
4. 外层 条件判断使用数组长度 - 1
5. 内层是 - 1 - i
6. 相邻元素进行比较 ,然后再进行两两交换

```
function bSort(arr)
  { var len =
  arr.length;
  for (var i = 0; i < len-1; i++) {
    for (var j = 0; j < len - 1 - i; j++) {
         //  相邻元素两两对比，元素交换，大的元素交换到后面
        if (arr[j] > arr[j + 1])
            {var temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
        }
    }
  }
  return arr;

//举个数组
myArr = [20,18,27,19,35];
//使用函数
bSort(myArr)
```

7. 判断数据类型的方式有哪些?每一个都有什么不同?(50%)

typeof 检测基本数据类型

instanceof 检测引用数据类型,但是无法确定哪一个引用数据的实例

Object.prototype.toString.call 检测引用数据类型

isArray 检测是不是一个数组

8. 如何改变函数内部的this指针的指向?(60%)

call apply bind

区别?

call和apply改变了函数的this上下文后便执行该函数,而bind则是返回改变了上下文后的一个函数。

他们俩之间的差别在于参数的区别，call和apply的第一个参数都是要改变上下文的对象，而call从第二个参数开始以参数列表的形式展现，apply则是把除了改变上下文对象的参数放在一个数组里面作为它的第二个参数。

9. js实现继承的方法有哪些?最少说出4种(50%)

原型链继承

构造函数继承

组合式继承

class类继承

10. 说一下什么是防抖?什么是节流?以及他们的应用场景?(必问)

防抖和节流的作用都是防止函数多次调用。区别在于，假设一个用户一直触发这个函数，且每次触发函数的间隔小于设置的时间，防抖的情况下只会调用一次，而节流的情况会每隔一定时间调用一次函数。

防抖:防抖(debounce): n秒内函数只会执行一次，如果n秒内高频事件再次被触发，则重新计算时间

- 每次 resize/scroll 触发统计事件
- 文本输入的验证（连续输入文字后发送 AJAX 请求进行验证，验证一次就好）

节流:节流(throttle): 高频事件在规定时间内只会执行一次，执行一次后，只有大于设定的执行周期后才会执行第二次。

- DOM 元素的拖拽功能实现（mousemove）
- 搜索联想（keyup）
- 滚动条滚动

11. js的数据类型有哪些?(30%)

6中 分别基本数据类型 和 引用数据类型

基本数据类型: number string boolean undefined null

引用数据类型: object object 和 array function

12. 说一下什么是宏任务?什么是微任务?(40%)

13. 说一下event loop?(40%)

Event loop(事件轮询)我们称之为事件循环机制. 说到event loop我们就必须先介绍单线程, 因为javascript是一个单线程语言,单线程就意味着，所有任务需要排队，前一个任务结束，才会执行后一个任务。如果前一个任务耗时很长，后一个任务就不得不一直等着。

这样下去,就会早场程序堵塞或者页面进入假死状态.

这个时候我们就需要使用event loop来进行解决,那么event loop核心是用来协调各种事件、用户交互、脚本执行、UI 渲染、网络请求等的一种机制. 实现的方法核心 通过定时器 + 消息队列 + 时间监听

14. 说一下数组常用的方法有哪些?(80%)

push

pop

shift

unshift

splice

concat

sort

substring

substr

slice

reserve

15. 说一下什么同源策略?以及解决跨域的方式有哪些?(必问)

所谓的同源策略指的是相同的协议、域名、端口号我们称之为同源策略,只要协议、域名、端口号有一个不一致的话就会产生跨域,而解决跨域的方式有跟多钟:

1. 使用cors 让 后端允许前端进行跨域操作
2. 使用nodejs做一层代理
3. jsonp
4. iframe
5. websocket

我们再项目当中最常用的方式再vue中进行跨域,vue中跨域需要再vue.config.js文件内进行配置

16. 说一下 localStorage、sessionStorage和cookie的区别?(必问)

> 首选呢,再html5没有出现之前,我们浏览器存储数据的方式一般都是采用cookie,如何要使用cookie需要对cookie进行二次封装才能够更加方便的去使用,使用coookie有一下的特点,cookie具有过期时间,到达指定的时间cookie就会消失,并且一个域名下最多只能存储20条cookie,并且cookie的大小有一定的限制,最后可存储4kb
>
> 而localstorage和sessionstorage都是html5新增的两个api方法,localstroage也称之为数据持久化,当我们使用localstorage将数据存储到本地的时候,如果不在浏览器上手动清楚或者不调用clear或者removeitem方法他是不会自动清楚的.
>
> 而sessionstorage我们称之为会话存储,使用sessionstorage存储的数据再关闭当前页面之后就会消失
>
> 他们和cookie最大的区别是 cookie有过期时间,而本地存储的两个方法如果不手动清除或者关闭浏览器就会一直存储,并且本地存储的api使用起来更加简洁和方便
>
> 以上就是我对本地存储和cookie的理解

17. 说一下从浏览器地址栏输入 URL 到页面渲染的整个流程?(80%)

> 1. DNS 解析
> 2. TCP 连接
> 3. 发送 HTTP 请求
> 4. 服务器处理请求并返回需要的数据
> 5. 浏览器解析渲染页面
> 6. 连接结束

四、小程序面试题

---

1. 小程序的生命周期?

小程序中的生命周期函数分为两类，分别是：

- 应用的生命周期函数
  特指小程序从启动 -> 运行 -> 销毁期间依次调用的那些函数
- 页面的生命周期函数
  特指小程序中，每个页面从加载-> 渲染 -> 销毁期间依次调用的那些函数

小程序的应用生命周期函数需要在 app.js 中进行声明。

```
App({
  //小程序初始化完成时，执行此函数，全局只触发一次。可以做一些初始化的工作。
  onLaunch: function(options) { },

  //小程序启动，或从后台进入前台显示时触发。
  onShow : function(options) { },

  //小程序从前台进入后台时触发。
  onHide : function() { }
})
```

小程序的页面生命周期函数需要在页面的 .js 文件中进行声明。

```
Page({
  //监听页面加载，一个页面只调用1次
  onLoad : function(options) { } ,
  //监听页面显示
  onShow : function() { },
  //监听页面初次渲染完成，一个页面只调用1次
  onReady : function() { },
  //监听页面隐藏
  onHide: function() { },
  监听页面卸载，一个页面只调用1次
  onUnload: function() { }
})
```

2. 小程序页面传参?

微信小程序的页面传参大致可分为以下四种：
1.url
2.storage
3.globalData

一、 首先，我们来说第一种。
这里我通过编程式导航跳转页面，并在url后面跟了参数

```
Wx.navigateTo({
url:  '/pages/me/me?id=22&title=个人中心'
})
```

我们在me页面中的onLoad生命周期函数中，传一个query就可以拿到传递过来的值。

```
onLoad(query){
const id = query.id
const title = query.title
}
```

二、 storage，这种方式和我们平常的Storage一样，不过它没有SessionStorage和LocalStorage之分,他都是永久缓存。但是它有同步，异步之分。

设置值：wx.setStorageSync(‘id’,22)

获取值：wx.getStorageSync(‘id’)

三、 这一种方式主要是通过全局变量globalData来传递

设置值：app.globalData.userid = 22

获取值：

const app = getApp()

const id = app.globalData.userid

3. 小程序授权登录?

```
wx.login({
    success: function(res){
      console.log(res)
      //获取登录的临时凭证
      var code = res.code;
      //将临时凭证发送给后端，后端通过微信登录开放接口获取到用户登录信息
      wx.request({
        url: 'https://www.usian.cn /wxLogin?code='+code,
        method: "get",
        success: function(result){
       //缓存信息便于其他页面使用
         wx.setStorageSync(‘userInfo’,result.userInfo)
         wx.redirectTo({
           url: '/pages/index/index'
         })
       }
```

```
    })
  }
})
```

4. 小程序分包加载?

分包指的是把一个完整的小程序项目，按照需求划分为不同的子包，在构建时打包成不同的分包，用户在使用时按需进行加载。

每个使用分包小程序必定含有一个主包。所谓的主包，即放置默认启动页面/TabBar 页面，以及一些所有分包都需用到公共资源/JS 脚本；

对小程序进行分包，可以优化小程序首次启动的下载时间，在多团队共同开发时可以更好的解耦协作。

分包后，小程序项目由 1 个主包 + 多个分包组成：

- 主包：一般只包含项目的启动页面或 TabBar 页面、以及所有分包都需要用到的一些公共资源
- 分包：只包含和当前分包有关的页面和私有资源

在小程序启动时，默认会下载主包并启动主包内页面，当用户进入分包内某个页面时，客户端会把对应分包下载下来，下载完成后再进行展示。

目前小程序分包大小有以下限制：

- 整个小程序所有分包大小不超过 20M
- 单个分包/主包大小不能超过 2M

开发者通过在 app.json **subpackages** 字段声明项目分包结构：

```
{
  "pages":[
    "pages/index",
    "pages/logs"
  ],
  "subpackages": [
    {
      "root": "packageA",
```

```
      "pages": [
        "pages/cat/cat",
        "pages/dog/dog"
      ]
    }, {
      "root": "packageB",
      "name": "pack2",
      "pages": [
        "pages/apple/apple",
        "pages/banana/banana"
      ]
    }
  ]
}
```

5. 小程序支付功能?

支付流程：

1.打开某小程序，点击直接下单。

2.wx.login获取用户临时登录凭证code，发送到后端服务器换取openId。

3.在下单时，小程序需要将购买的商品Id，商品数量，以及用户的openId传送到服务器。

4.服务器在接收到商品Id、商品数量、openId后，生成服务期订单数据，同时经过一定的签名算法，向微信支付发送请求，获取预付单信息(prepay_id)，

同时将获取的数据再次进行相应规则的签名，向小程序端响应必要的信息。

5.小程序端在获取对应的参数后，调用wx.requestPayment()发起微信支付，唤醒支付工作台，进行支付。

6.接下来的一些列操作都是由用户来操作的包括了微信支付密码，指纹等验证，确认支付之后执行鉴权调起支付。

7.鉴权调起支付：在微信后台进行鉴权，微信后台直接返回给前端支付的结果，前端收到返回数据后对支付结果进行展示。

8.推送支付结果：微信后台在给前端返回支付的结果后，也会向后台也返回一个支付结果，后台通过这个支付结果来更新订单的状态。

其中后端响应数据必要的信息则是wx.requestPayment方法所需要的参数，大致如下：

```
wx.requestPayment({
  // 时间戳
  timeStamp: '',
  // 随机字符串
  nonceStr: '',
  // 统一下单接口返回的 prepay_id 参数值
  package: '',
  // 签名类型
  signType: '',
  // 签名
  paySign: '',
  // 调用成功回调
  success () {},
  // 失败回调
  fail () {},
  // 接口调用结束回调
  complete () {}
})
```

6. 小程序组件通信?

组件通信就是：子传父，父传子，兄弟传值

1. 首先来说说父传子

小程序中的父传子和 vue 中的父传子很相似

父组件中引入子组件，给子组件添加属性

```
<view>
        <Child title="好好学习，天天向上"/>
    </view>
```

在子组件的js文件的properties中接受传过来的值

```
properties: {
```

```
    title:String
  }
```

2. 接下来说子传父

小程序的子传父和 vue 也很相似，接下来我们说说具体步骤

在子组件中触发一个事件

```
<view>
  <button bindtap="toFatherevent">
    点我把子组件的值传给父组件
  </button>
</view>
```

接下来在子组件的js中对应的toFatherevent上添加派发的事件和要向父级传递的数据

```
methods: {
  toFatherevent(){
    this.triggerEvent('tofather','我想把我的值传给我的爸爸')
  },
}
```

父组件监听子组件派发过来的事件和要接收的数据

```
<view>
        <Child  bindtofather="giveme" />
    </view>
```

在父组件的js中通过giveme事件获取数据

```
giveme(e){
 console.log(e.detail) //在控制台输出我们想接收的值
 this.setData({
  title:e.detail  //更新数据
 })
},
```

3. 最后来思考兄弟传值

兄弟传值：就是前两者的结合

假如两个子组件为 A1,A2，另一个父组件为 B

让组件 A1 利用子传父将值传给 B，具体步骤如上
B 接收 A1 传过来的值，然后利用父传子将值传给 A2，具体步骤如上
最后在 A2 中接收即可

五、性能优化面试题

---

webpack的性能优化

```
1
```

常规的性能优化

```
1  减少http请求：
2  减小资源体积
3  缓存：可以通过以下几个方面来描述：
4     DNS缓存
5     CDN部署与缓存
6     http缓存
```

六、招聘平台注册与简历投递

6.1 注册招聘平台

主推:

boss直聘 : 新用户不能注册

智联招聘 ： 每天最多可以投150家

其他平台:

前程无

忧拉钩

直聘猎

聘网

6.2 投递简历的注意事项

早 : 9点左右开始投

递简历中: 2点开始投

递简历

晚: 睡觉之前打开boss,然后投递几家

6.3 投递简历的流程

先投递智联招聘,一键投递,一次可以投30家,一共可以投递5次, 大概算下来一共是可以投递150家,

只要三四分钟就能投完接下来再boss直聘开始boss聊天、约面试题, 不要看招聘需求,看完招聘需

求,你就会觉得你干不了,或者不符合要求.

- 注意事项
  - 面试官问到你的身份证年龄和实际年龄一致
  - 吗? 不一致你的学历是学习网可查吗? 民教网
- 每天要完成的任务
  - boss直聘的沟通量必须至少达到200家

6.4 找工作阶段的事项

- 早上7点半的时间到教室,背面试题,总结前一
- 天的面试题晚上10点回宿舍
- 把o"er发给老师,通过老师的批准就可以不用按时来学校了
- 如果是线下的面试的同学,面完一家就和老师电话或
- 微信沟通一下所有人晚上如果没有面试的话,必须得在教室
- 面试的时候必须录音,面试结束第一时间把录
-

音发给老师做一个面试排期表,将每天的面试排期表发到群里面