[一、vue.H5 1](#_Toc3630)

[学习： 1](#_Toc29749)

[1. js学习 1](#_Toc2635)

[1.技术点 2](#_Toc18744)

[1. 输入验证 2](#_Toc15312)

[（1. 小数点后两位 2](#_Toc32025)

[2. 点击旋转过渡 刷新图标 2](#_Toc14465)

[3. vue的一些动画 3](#_Toc26935)

[3. vue. 取元素中的value值？？？ 3](#_Toc19511)

[4. 连续点击控制 重复点击 4](#_Toc10747)

[5. 强制刷新，只刷新渲染数据 4](#_Toc2514)

[6 图片预览 preview 4](#_Toc5782)

[2.功能模块 5](#_Toc25694)

[1.群聊 私聊——strophe.js 5](#_Toc29692)

[1）思路 5](#_Toc6833)

[2.单聊程序步骤：（对比群聊） 10](#_Toc9363)

[3.群聊程序步骤：（超时处理/检测、重连/重发...） 14](#_Toc20700)

[4. 发送图片：（聊天室、openfire） 16](#_Toc25188)

[3.vue常用知识点 19](#_Toc19989)

[1.路径 19](#_Toc16342)

[2. 调store中的方法 直接重定向方式 19](#_Toc19210)

# 一、vue.H5

## 学习：

### js学习

收录项目封装的公共函数：一般放在utils

## 1.技术点

### 1. 输入验证

#### （1. 小数点后两位

0.搜百度：

1.移动端： @input=’changeAmount’

changeAmount(e){

let value=e.target.value

If(value<0){

value=’’

}else if(!/^\d+(\.d+)?$/.test(value){

value=value.toString().match(/^\d+(?:\.\d{0,2})?/)[0]

}

this.amount=value

e.target.value=value

}

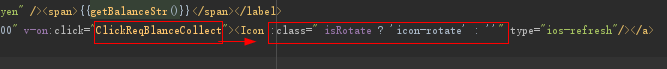
1. pc端做法：

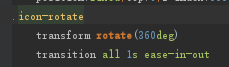


### 2. 点击旋转过渡 刷新图标



做class——方法切换class



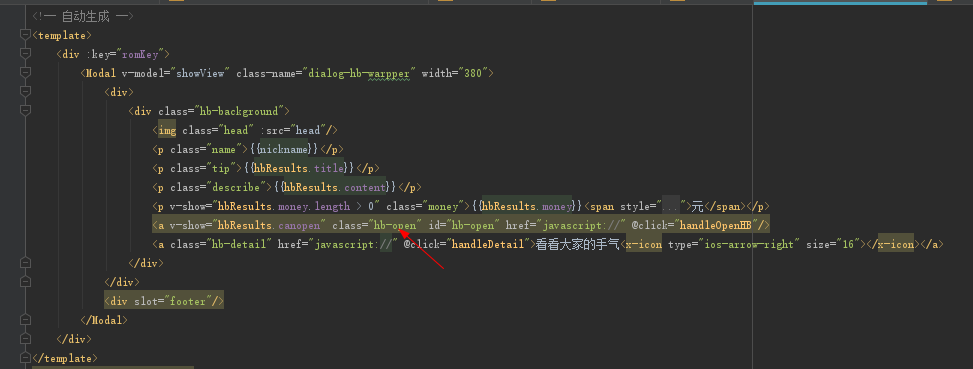


### 3. vue的一些动画

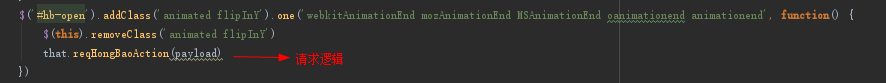
功能：点击/元素翻转



元素：

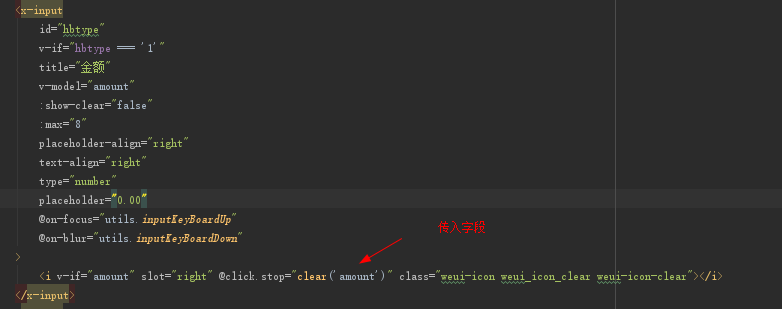


js动画逻辑：

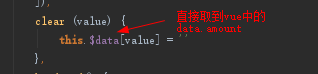


### 3. vue. 取元素中的value值？？？

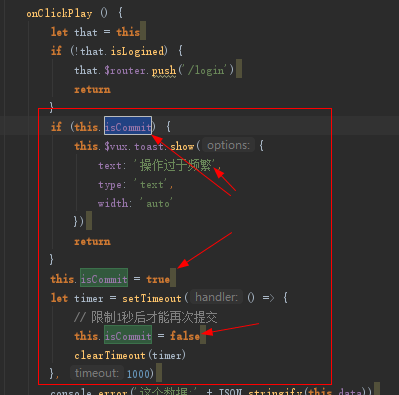
1.传入字段



2.改变对应的data



### 4. 连续点击控制 重复点击



### 5. 强制刷新，只刷新渲染数据

（数据嵌套太深，diferent有可能检测不到数据变动）

**this**.$forceUpdate()

### 6 图片预览 preview





启用vue预览：数据变动时加：this.$previewRefresh && this.$previewRefresh()

### 7.右键弹出框 撤回

## 2.功能模块

### 1.群聊 私聊——strophe.js

<https://xmpp.org/extensions/> 当时用的是**XEP-0013**

官方文档：<http://strophe.im/strophejs/>

参考：<https://www.cnblogs.com/wucao/p/4223169.html>

#### 1）思路

引入strophe.js，

创建房间类及其方法，

拼到对应数据后，引用strophe的API链接服务器。

1. 群聊发消息：
2. ComChatInput2：

群主：

1. this.sendMucText({ room\_name: this.chat\_key, body: this.inputText })

参数：房名，输入内容

群员：

1. this.sendWarnText({ room\_name: this.chat\_key, body: '不能发言，群主开启禁聊' })

参数：房名，提示内容

1. **this**.sendMucText({  
    room\_name: **this**.chat\_key,  
    body: **this**.inputText,  
    timeoutCallback: (uuid) => {  
    **let** result = that.checkCanChatForMe()  
    **return** result  
    }  
   })

参数：房名，输入内容，回调：。。。。

1. **this**.sendWarnText({ room\_name: **this**.chat\_key, body: '没有发言权限，请联系群主，如已解除禁言，请重进聊天室' })

参数：房名，输入内容

2. actionsXmpp.js (store.action):

sendMucText(context, payload) {  
 **let** room = RoomMgr.*getRoom*(payload.room\_name, ConnMgr.*getConn*(context, payload.resourceTag))  
 room.sendMessage({ //romm:返回来的conn（strophe请求对象，看下面的room）  
 memeberNick: payload.memeberNick,  
 cmd: MessageCmd.CLIENT\_CMD\_MESSAGE,  
 body: { text: payload.body },  
 uuid: payload.uuid,  
 callback: payload.callback,  
 timeoutCallback: payload.timeoutCallback  
 })  
},

入参：上下文、房名，输入内容

2.1. room （房间类）：

1. 创建房间类（群聊类）：

**let** room = RoomMgr.*getRoom*(payload.room\_name, ConnMgr.*getConn*(context, payload.resourceTag))

RoomMgr:*getRoom*(name, conn)//返回：room（群聊类）

入参：房名、getConn(strophe请求对象配置)

1. rooms：已存在的房间集合：（是保存在store的）

actions——actionsXmpp——RoomMgr里——

1. custCode:user编码：1\_c\_lemon

2.1集合中没有历史房间：

room = **new** RoomClient(name, conn)

入参：名字，strophe类

★给room创建**关键方法类 ——群聊类**

1.1 ConnMgr.getConn(方法，有引入strophe): 返回：conn（strophe请求对象）：

*getConn*(context, resourceTag = **null**)

1. this.scheduler()：如果已有conn(strophe类),每5s发送一次ping包到服务器
2. 获取信息：
   1. outXmpp

租户信息.后台返回的xmpp地址或者端口号（拼下面url用）：xmpp\_outer （tenant/getters .tXmppOuter）

* 1. custCode 编号？（user userInfo.cust\_code)
  2. pass 密码：userInfo.password1st
  3. hashCustCode 上面3个的加密
  4. jid getClientJid（用user\_info和时间戳拼房间号）：房间号

1. 拼url：判断http协议是否为http:开头

**4.this**.*conn* = **new** Strophe.Connection(url)

5. 请求ping包 this.conn.ping.addPingHandler

6. *conn*.addHandler(  
 (msg) => {  
 **const** jsonMsg = MessageParse(msg)  
 **this**.handleMessage({jsonMsg, local: **false**})  
 **return true** },  
 **null**,  
 'message',  
 **null**,  
 **null**,  
 **null**)

（异步请求服务器是否有新信息（房间号、信息之类）

1. strophe重连线路判断

2. 群聊类：RoomClient / 群聊类方法：（发送信息）

sendMessage(payload) {

入参：（payload：actionsXmpp.js 方法：sendMucText)

cmd：消息类型 （普通消息）

body：{text：this.inputText }

timeoutCallback：超时后回调，拿group\_id查后台是否有权限发信息

1.拼接msg：**疑问**：msg.c('body', null, JSON.stringify(sendBody))

2.**this**.connection.send(msg)// 发送消息给服务器

3.**this**.sendLocalMsg(msg)//本地保存，不走服务器

onMessage(xml, { name: this.to }, localMsg.body.state !== 999)

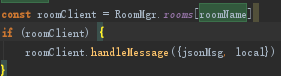
//xml消息转json

入参：xml信息，房名+ ['@conference.'+租户的xmpp](mailto:'@conference.'+租户的xmpp)，发送状态：1 发送中 2 发送超时 3 隐藏消息 999 发送成功

ConnMgr.handleMessage({jsonMsg, local}) //处理消息到本地保存

入参：json消息，消息发送状态

1. 拿到房间号：roomName
2. 定位到rooms中的房间，再执行roomClient的（handleMessage）的方法



入参：json消息，消息发送状态

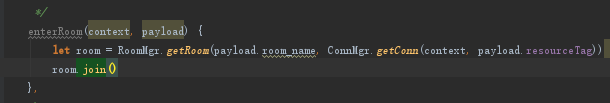
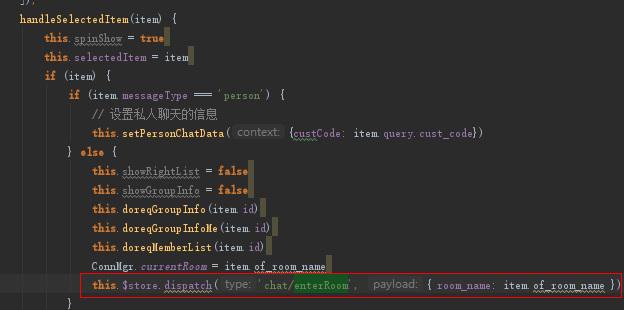
1.分析消息，做出提示

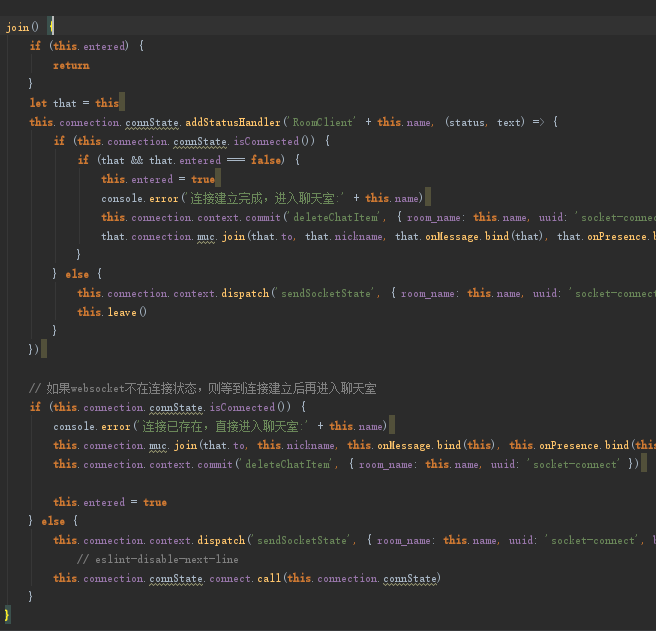
2. store保存：各种发送状态，消息

**this**.connection.context.commit('addChatItem', { key: **this**.name, item: jsonMsg })

入参：房间名，jsonMsg

1. 群聊自动接收：





群聊类.join方法——判断房间是否存在—不—删除聊天记录——

that.connection.muc.join(that.to, that.nickname, that.onMessage.bind(that), that.onPresence.bind(that))

相当于——new Strophe.Connection(url). muc.join(入参)

入参：1.房名+ ['@conference.'+租户的xmpp](mailto:'@conference.'+租户的xmpp)

2.用户昵称

3.that.onMessage.bind(that)//接收信息并处理信息的方法——绑定给群聊实例——传给strophe

onMessage入参（shrophe会给）：

xml

房名+ ['@conference.'+租户的xmpp](mailto:'@conference.'+租户的xmpp)

消息状态

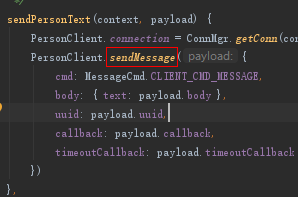
4.that.onPresence.bind(that)//

入参：xml信息，房名+ ['@conference.'+租户的xmpp](mailto:'@conference.'+租户的xmpp) (this.to)

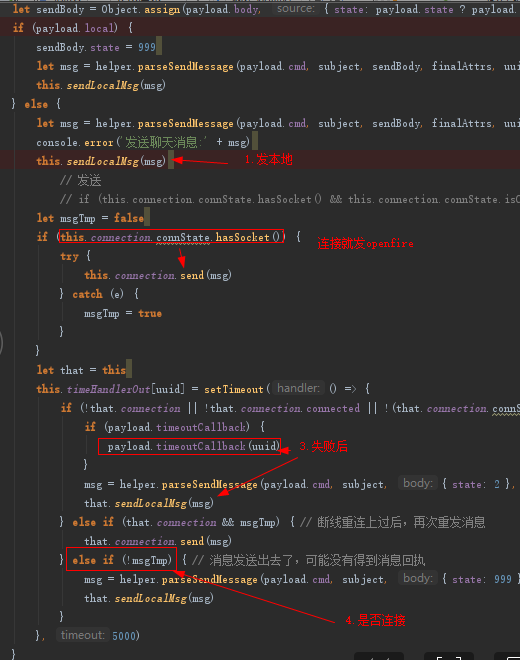
1. 私聊发送消息：
2. 单聊列表personChatItem来源：store+心跳信息
3. addPersonChatItem
4. PersonClient.js/handleMessage
5. ConnMgr.js/handleMessage
6. PersonClient.js/onMessage

### 2.单聊程序步骤：（对比群聊）

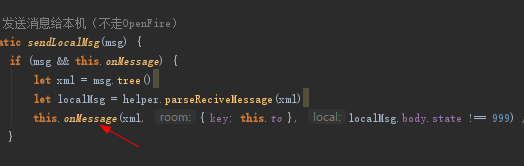
1.



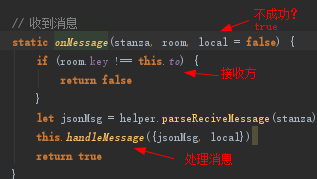
2.



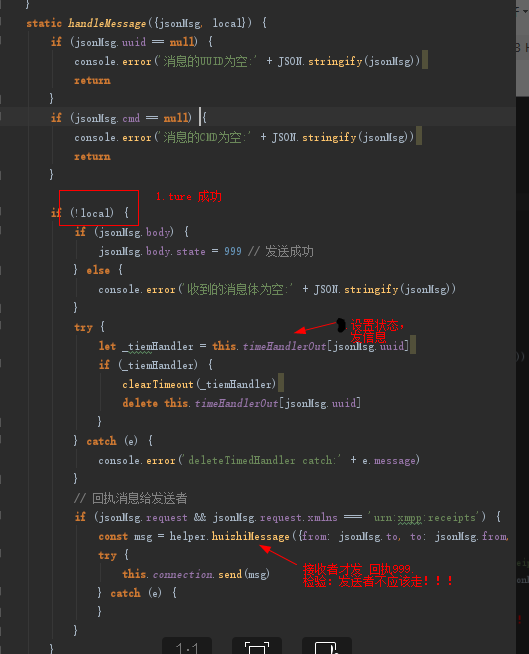
3.



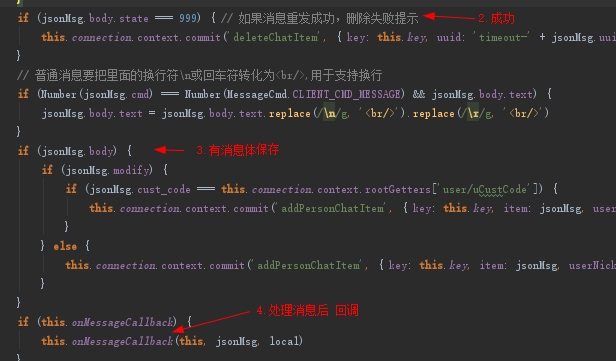
4.



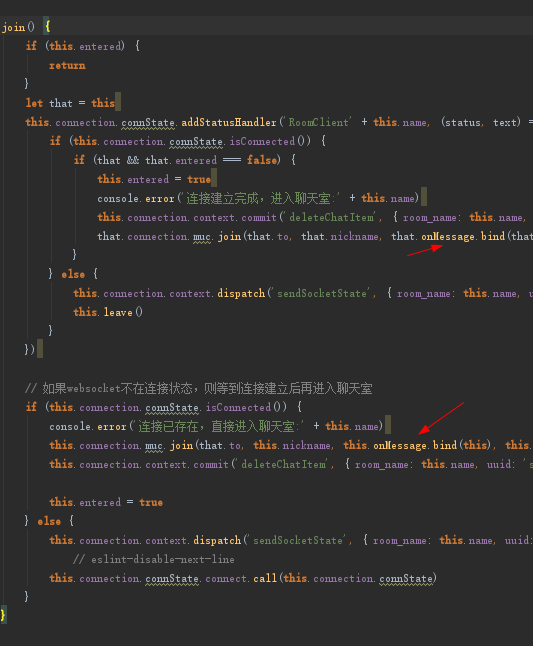
5.



5.

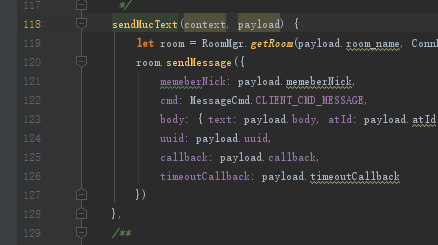


5.1群聊 进入聊天室程序

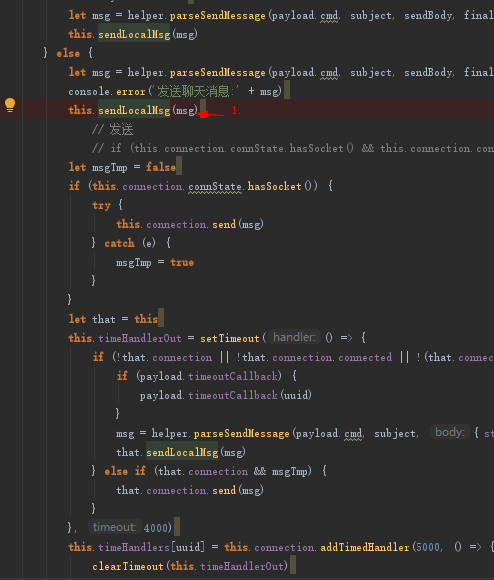


### 3.群聊程序步骤：（超时处理/检测、重连/重发...）

1.



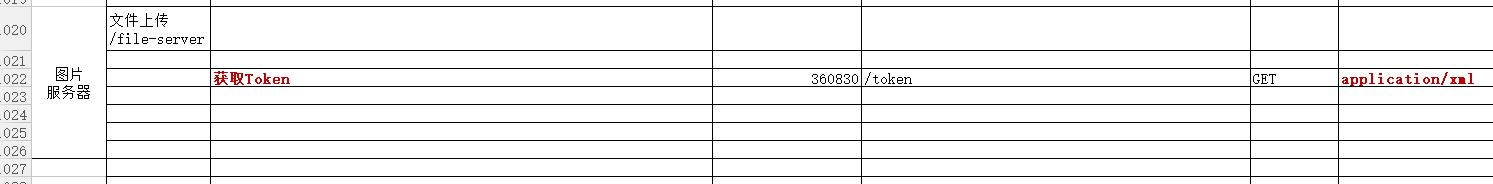
2.

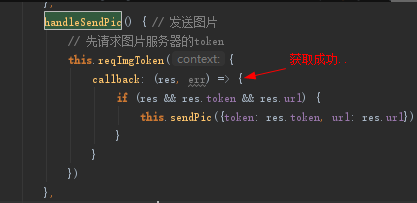


### 4. 发送图片：（聊天室、openfire）

1.点击发送图片按钮

2.接口文档：获取图片服务器token和url

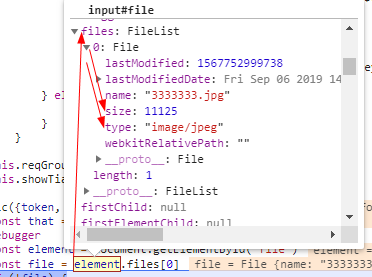




3. 查看input中图片的信息。

①~⑦验证、压缩、上传、成功后发openfire信息（url）

sendPic({token, url}) {  
 **const** that = **this  
 debugger  
 const** element = document.getElementById('file')  
 **const** file = element.files[0] ①

信息：  
 **if** (!file) {  
 **return** }  
 element.value = '' // 清空file的值，避免取消事件  
 */\*\*  
 \* 加了111，是因为在打包过后他们不相等了  
 \*/* **if** (!(`${file.type}111` === 'image/png111' || `${file.type}111` === 'image/jpg111' || `${file.type}111` === 'image/jpeg111' || `${file.type}111` === 'image/webp111')) { // 检测是否是图片格式 png、jpg、webp  
 **this**.$Message.error('请重新选择图片，支持png、jpg、webp格式')  
 **return** }  
 **const** tmpFunc = **function** (res) {  
 that.uploadImg({//⑤引入该api  
 file: res,  
 token,  
 url,  
 callback: (res, err) => {  
 console.warn('bebedon 上传图片的返回===>', res, err)  
 **if** (res && res.url) { // ⑥成功上传完至图片服务器了  
 **let** imgUrl = res.url  
 **if** (!/^http/.test(imgUrl)) {  
 imgUrl = `${location.protocol}//${imgUrl}` // 与域名相同  
 }

//⑦开始发送openfire消息  
 that.handleSendMessage({type: 'image', imgUrl})  
 } **else** {  
 that.$Message.error('上传图片失败')  
 }  
 }  
 })  
 }  
 **if** (file.size > 2 \* 1024 \* 1024) { // ②图片超过2M不给与发送  
 that.$Message.error('请选择2M以下的图片')  
 **return** }  
 **let** ratio = 1  
 **if** (file.size > 1024 \* 1024) { // 一张图片大于1M的时候压缩到原来的一半  
 ratio = 0.5  
 }

//③压缩图片  
 utils.*compressFile*(file, {ratio}, (res) => {  
 tmpFunc(res)//④上传图片  
 })  
},

③：

// 压缩图片  
**static** *compressFile*(file, {ratio}, callback) {  
 **let** fileObj = file  
 **let** reader = **new** FileReader()  
 reader.readAsDataURL(fileObj) // 转base64  
 reader.onload = **function**(e) {  
 **let** image = **new** Image() // 新建一个img标签（还没嵌入DOM节点)  
 image.src = e.target.result  
 image.onload = **function** () {  
 **let** canvas = document.createElement('canvas')  
 **let** context = canvas.getContext('2d')  
 **let** imageWidth = image.width \* ratio  
 **let** imageHeight = image.height \* ratio  
 **let** data = ''  
 canvas.width = imageWidth  
 canvas.height = imageHeight  
 context.drawImage(image, 0, 0, imageWidth, imageHeight)  
 data = canvas.toDataURL('image/jpeg') // 输出压缩后的base64  
 **let** arr = data.split(',')  
 **let** mime = arr[0].match(/:(.\*?);/)[1] // 转成blob  
 **let** bstr = atob(arr[1])  
 **let** n = bstr.length  
 **let** u8arr = **new** Uint8Array(n)  
 **while** (n--) {  
 u8arr[n] = bstr.charCodeAt(n)  
 }  
 **const** lastIndex = fileObj.name.lastIndexOf('.')  
 **const** newName = fileObj.name.slice(0, lastIndex)  
 **let** files = **null  
 try** {  
 // 这儿存在兼容性问题  
 files = **new** window.File([**new** Blob([u8arr], {type: mime})], `${newName}.jpeg`, {type: 'image/jpeg'}) // 转成file  
 } **catch** (e) {  
 files = file  
 }  
 callback(files) // 回调  
 }  
 }  
}

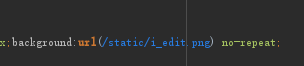
⑤：

uploadImg(context, {file, token, url, callback}) {  
 **const** param = **new** FormData()  
 param.append('file', file, file.name)  
 param.append('token', token)//①拼装formData数据  
 **const** config = {  
 headers: {  
 'Content-Type': 'multipart/form-data'  
 }  
 }  
 **if** (callback) {  
 axios.post(url, param, config)//①服务器的url token、文件数据、请求头  
 .then(res => {  
 callback(res.data, **null**)  
 })  
 .catch(e => {  
 callback(**null**, e)  
 })  
 .finally(res => { // 请求完成后，可能需要的操作  
 // console.warn('bebedon uploadImg finally===>', res)  
 })  
 } **else** {  
 **return** axios.post(url, param, config)  
 }  
}

## 3.vue常用知识点

### 1.路径

1.打包 默认静态路径 绝对路径



### 2. 调store中的方法 直接重定向方式



## 4. vue-app

### 1.页面跳转

