GitHub账号：[448919239@qq.com](mailto:448919239@qq.com)

1.vue中的跳转

1.直接点击跳转 this.$router.push({path: '/home/first'})

2.链接跳转 <router-link to="/about">About</router-link>

3.传参跳转 this.$router.push({path: '/home/first'})

2.点击一项当前项颜色变红，其他项颜色变灰

// 模板视图层

<template>

<div class="hello">

<ul>

<li

v-for="(item, index) in list"

:class="{'active':ind === index}"

@click="changeBgc(index)">{{item}}</li>

</ul>

</div>

</template>

// 数据层

data () {

return {

list: ['苹果', '香蕉', '菠萝', '芒果'],

ind: ''

}

},

// 逻辑层

methods: {

changeBgc: function (index) {

this.ind = index

}

}

3.vue-resource post设置{emulateJSON:true}

启用该选项后，请求会以application/x-www-form-urlencoded作为MIME type，就像普通的HTML表单一样。form形式传给后台，而非json形式传递给后台

4.vue中带接口名称的数据请求

var requestdata = JSON.stringify({

getOrderSingleList: {

entity: this.onename,

staffcode: 'njpsy01',

destatictime: this.date

}

});

this.$http.post(this.baseurl, requestdata, { emulateJSON: true }).then()

5.value中的数据绑定

v-model:value="Receivedeliveryno" // Receivedeliveryno是变量

6. vue利用type=file实现本地上传图片

https://blog.csdn.net/guoyiq/article/details/79395249

当然也可以用自己的方法：

<input class="change" @change="uploadInput($event)" type="file" multiple="multiple" />

方法部分：

uploadInput(e){

var that=this;

var oMyForm = new window.FormData();

oMyForm.append("file",e.target.files[0]);

this.$vux.loading.show({

text: "图片正在上传..."

})

// 选中图片点击“上传”按钮需要向服务器发出请求，从而得到返回地址

this.$http.post("http://files.scn.weilian.cn/upload",oMyForm,{emulateJSON:true}).then((response) => {

// 查看服务器端给到我的数据

console.log(response.data);

that.successmsg = "图片上传成功";

that.show2 = true;

that.$vux.loading.hide();

// 把上传图片返回的地址存储到本地

that.imgURL = response.data.downloadPath;

}, (response) => {

that.errormsg = "图片上传失败";

that.show1 = true;

that.$vux.loading.hide();

});

},

div弹框部分

<toast v-model="show1" :time="toasttime" position="middle" type="cancel">{{errormsg}}</toast>

<toast v-model="show2" :time="toasttime" position="middle" type="success">{{successmsg}}</toast>

7. flex经典布局教程

基础篇：http://www.ruanyifeng.com/blog/2015/07/flex-grammar.html?^%$

实例篇：http://www.ruanyifeng.com/blog/2015/07/flex-examples.html

7. 监听路由的变化

https://blog.csdn.net/firebird\_one/article/details/79477659 （学习参考）

具体用法：

watch:{

'$route':function(to,from){

console.log('观察路由的变化',"to",to.path)

if(to.path.slice(1)=="abnormalReceive"){

this.titletext="异常签收"

}

if(to.path.slice(1)=="waybill"){

this.titletext="订单详情"

}

},

result:function(value) {

}

}

8. :is的用法

is只能用于绑定组件

9. 项目中用到的地图学习

http://lbs.amap.com/api/javascript-api/reference/map/

10. VUX的使用教程

https://blog.csdn.net/revival\_liang/article/details/78267992

11. 最基本的Vuex计数应用

参考地址：https://jsfiddle.net/n9jmu5v7/1269/

或者参考地址：Vuex官网——开始——最基本的Vuex计数应用（绿色文字）

HTML结果如下：

<div id="app">

<p>{{ count }}</p>

<p>

<button @click="increment">+</button>

<button @click="decrement">-</button>

</p>

</div>

逻辑部分：

// make sure to call Vue.use(Vuex) if using a module system

const store = new Vuex.Store({

state: {

count: 0

},

mutations: {

increment: state => state.count++,

decrement: state => state.count--

}

})

new Vue({

el: '#app',

computed: {

count () {

return store.state.count

}

},

methods: {

increment () {

store.commit('increment')

},

decrement () {

store.commit('decrement')

}

}

})

13. vue-amap基于Vue 2.x与高德的地图组件应用

https://elemefe.github.io/vue-amap/#/

https://elemefe.github.io/vue-amap/#/zh-cn/introduction/install

http://lbs.amap.com/api/javascript-api/example/map/map-show（要结合高德开放平台）

可以用的高德地图密钥：

a5675607fead313203a1675be72fa182

d345d0dc5ccaa689e39666a327629106

73358a8e68d85587063bebce3091fd66

14. IntelliJ IDEA 2016.1(64)提交代码工具的使用

1.先打开IntelliJ IDEA 2016.1(64)——打开VCS——Checkout from Version Control把代码给拉下来——打开拉下来的代码——Local Changes ——Add to Vcs—— commit and push 注意一定要切换分支，提交到开发环境

15. 好用的表格网站学习

1.效果图

http://panjiachen.github.io/vue-element-admin/#/dashboard

2.源码

https://github.com/PanJiaChen/vue-element-admin

16. axios请求的学习

1.axios是一个基于Promise的http库，可以用在浏览器和node.js中

参考学习地址1：https://www.jianshu.com/p/df464b26ae58

参考学习地址2：https://www.cnblogs.com/rogerwu/p/7610478.html

17. 表格排序的问题

https://www.cnblogs.com/lymblog/p/7458941.html

18. 只保留相机拍照的功能

<input class="change" @change="uploadInput($event)" type="file" multiple="multiple" capture="camera" accept="image/\*" />

19. 如何将非高德地图转换为高德坐标系？

http://lbsbbs.amap.com/forum.php?mod=viewthread&tid=724&highlight=%E5%9D%90%E6%A0%87%E8%BD%AC%E6%8D%A2

20. 绑定事件的相关知识：

window.addEventListener("popstate", function (e){

alert('我监听到了浏览器的返回按钮事件啦');

}, false);

这里的false代表的是冒泡阶段调用，如果改为true则表示捕获阶段调用

20. H5中JS禁用安卓手机物理返回键，微信浏览器中也支持，直接在主要逻辑处引用以下代码即可

XBack = {};

(function(XBack) {

XBack.STATE = 'x - back';

XBack.element;

XBack.onPopState = function(event) {

event.state === XBack.STATE && XBack.fire();

XBack.record(XBack.STATE); //初始化事件时，push一下

};

XBack.record = function(state) {

history.pushState(state, null, location.href);

};

XBack.fire = function() {

var event = document.createEvent('Events');

event.initEvent(XBack.STATE, false, false);

XBack.element.dispatchEvent(event);

};

XBack.listen = function(listener) {

XBack.element.addEventListener(XBack.STATE, listener, false);

};

XBack.init = function() {

XBack.element = document.createElement('span');

window.addEventListener('popstate', XBack.onPopState);

XBack.record(XBack.STATE);

};

})(XBack); // 引入这段js文件

XBack.init();

XBack.listen(function() {});

20. 低版本手机空白，内容无法显示的问题

安装：

npm install babel-polyfill --save 和npm install es6-promise --save

main.js中引入：

import 'babel-polyfill';

require('es6-promise').polyfill();

21. 前端解决跨域的问题和方法

1.使用JSONP的形式解决

2.在用原生的ajax或者jQuery中的$.ajax() 方法请求时设置请求头

Access-Control-Allow-Headers:x-requested-with,content-type,sessionid

Access-Control-Allow-Methods:GET,POST,HEAD,PUT,DELETE,OPTIONS

Access-Control-Allow-Origin:\* (当然后端设置这个请求头比较方便)

// 这里是前端设置的请求的方法

function makeRequest() {

alert("inside makeRequest()");

var settings = {

type: "GET",

url:URL+"?"+REQUEST,

dataType:"xml",

error: function(XHR,textStatus,errorThrown) {

alert(XHR,textStatus,errorThrown);

},

success: function(data,textStatus) {

$("body").append(data);

},

headers: {

"Access-Control-Allow-Origin": "\*",

"Access-Control-Allow-Headers": "X-Requested-With"

}

};

$.ajax(settings);

}

22. 正则表达式的使用

1.test()的使用

字符串在括号内

var regString = /Y[0-9]{14}-[0-9]{5}/;

var testString = 'Y20180813000004-00001';

console.log('检测正则表达式', regString.test(testString));

2.match()的使用

字符串在括号外

第一种情况

var reg = new RegExp("Y[0-9]{14}-[0-9]{4}");

var testString = 'Y20180813000004-00001';

var regArr = testString.match(reg);

console.log('查看正则表达式的检测', regArr);

第二种情况

var reg = new RegExp("(^|&)"+ 'sessionId' +"=([^&]\*)(&|$)");

var nameIdString = 'sessionId=123456';

var regArr = nameIdString.match(reg);

console.log('查看正则表达式的检测', regArr);

22. vue中的请求方式：

post的第一种请求状态:

var objParam={

"updateAppExceptionOrder":{ // 这里是方法名

"id":this.$route.query.id,//交接单号

"abnormalreason":this.txt, //异常原因

"count":"", //拉回数量

"imageurl":this.imgURL, //上传图片的地址

"exception":this.reasonValue, //异常内容

"userName":this.personID.user.userName

}

};

this.$http.post(this.baseurl, JSON.stringify(objParam), {emulateJSON: true}).then((response) => {

console.log('请求成功', response);

}).catch((response) => {

console.log('请求失败', response)

})

post的第二种请求状况

var driverIdRequest = {

"condition":"",

"driverid" :this.personID.user.account,

"jointime" :''

};

var joinBaseurl = this.api + 'api/getSignDataListForApp';

this.$http.post(joinBaseurl, driverIdRequest, {emulateJSON:true}).then((response) => {

console.log('请求成功', response);

}).catch((response) => {

console.log('服务器错误！！！', response);

});

get请求状态:把参数都要拼接到url上

var joinBaseurl = that.baseurl + 'api/scanGoodsTag';

var useScanQRCodeRequest = '';

useScanQRCodeRequest = joinBaseurl + '?' + 'logisticsno=' + that.inputSearchText.substr(0, 15) + '&disposeids=' + breakNumber + '&sn=' + that.inputSearchText.substr(16) + '&carloadingno=' + that.$route.query.carloadingno + '&sessionId=' + that.UserSessionId + '&isContinuityScan=' + '';

that.$http.get(useScanQRCodeRequest, {emulateJSON: true}).then((response) => {

console.log('请求成功', response);

}).catch((response) => {

console.log('请求失败', response)

});

23. vuex的简单使用原理：

可以在main.js中也可以在仓库store文件中输入如下内容：

（这里在main.js中为例）

在main.js中引入vuex

import Vuex from 'vuex';

Vue.use(Vuex);

const store = new Vuex.Store({

state: {

username: ""

},

mutations: {

changenameFn(state, newnameParam) {

state.username = newnameParam;

}

}

});

在A组件中提交一个函数和值，去改变组件仓库里面的值

this.$store.commit("changenameFn", response.data.data[0].userName);

在B组件中接收这一个值:

首先在B组件中引入vuex的方法

import {mapState} from 'vuex';

在B组件computed中接收

computed: {

...mapState(['username'])

}

24. vue中有关input的一些事件

<input v-model="item.amount" min="0" @input="inputFunc($event, index)" @focus="inputFocus($event, index)" @keyup.enter="inputOnSearch" />

25. vue中computed中的一些经典的用法

<template>

<div>

<div class="submitTask" :class="submitStatus" @click="sublitmission">提交任务</div>

</div>

</template>

computed: {

// 用来控制提交任务的背景色

submitStatus() {

if (this.canSubmitTask === 0) {

return 'no-submit';

}else if (this.canSubmitTask > 0) {

return 'can-submit';

}

}

}

其中:class还有另外两种绑定类名的方法

:class="{'active':ind === index}"

:class="[clickable? 'canClick' : 'noCanclick']"

26.vue的调试工具vue-devtools的使用

1. git clone https://github.com/vuejs/vue-devtools

2. cd vue-devtools

3. npm install

4. 修改manifest.json文件 vue-devtools/shells/chrome/manifest.json中的

"background": {

"scripts": [

"build/background.js"

],

"persistent": false

}里面的persistent修改为true

5. npm run build

6. 扩展Chrome插件

Chrome浏览器 >  更多程序 > 拓展程序 > 选中开发者模式

点击加载已解压程序按钮, 选择 vue-devtools > shells > chrome 放入, 安装成功

27.vue的input框使用v-model时，需要复制一份数据，为了防止复制的数据随着本身循环的数据的改变而改变使用的方法：

that.Data = response.data.data[0].list;

that.copyData=JSON.parse(JSON.stringify(response.data.data[0].list));

(先转移为字符串再转化为数组就好了)

28.转换为dataURL时方便的压缩

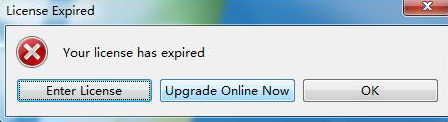
var dataurl = canvas.toDataURL('image/png');

var dataurl2 = canvas.toDataURL('image/jpeg', 0.8);

29.开发用到的工具

IntelliJ IDEA 2016.1(64)

TortoiseSVN(64Bit)

如果出现了这个就用如下的地址破解 http://idea.imsxm.com 步骤：点击Enter License

30.原始套路开发解决子元素DOM还没加载出来，就需要绑定事件的问题

$("#filelist").on('click', ".video\_link",function(){

Console.info("run here");

});

解释说明：#filelist是父级元素， .video\_link是即将要生成的DOM

31.今后拉取代码及提交代码时，都要留意是在主分支上，还是在分分支上

例如：gitLab上建一个主的master分支。然后有两个A，B分支.在IntelliJ IDEA 2016.1(64)右下角，Remote Branches处选择Checkout as new Local branch建立对应的本地分支

32.微信支付宝的支付功能

web页面调用支付宝支付：https://blog.csdn.net/weixin\_41003211/article/details/78541735

微信支付：https://www.jianshu.com/p/d04ec2891b1b

待解决提高的地方

1.vue打包好生成的dist文件怎么部署到服务器上？

怎么部署啊怎么部署啊？？？？？？？？？？？？？？？还有待提高

2.工作中怎么使用远程工具

向日葵，TeamViewer

3.如何使用gitLab进行代码的管理

还在学习中

4.如何将npm run build好的dist文件打包成APK,IOS的安装包？

https://blog.csdn.net/qqxinxi/article/details/71401574

5.jquery的深入学习提升

\*\*\*\*\*\*\*\*\*\*\*\*

6.js及ES6的深入学习提升

\*\*\*\*\*\*\*\*\*\*\*

7.react的深入的学习

\*\*\*\*\*\*\*\*\*\*\*

**//再谈前后端分离式 手把手从0打造电商平台-用户操作系统的学习——代码要敲一遍**

**//再谈前后端分离式 手把手从0打造电商平台-后台管理系统的学习——代码要敲一遍**

**//react的学习**

8.小能在线客服开发文档

**http://doc3.xiaoneng.cn/**

**账号为：iaw\_lg**

**密码为：ntalker123**

9.关于验证码图片的问题

**<cite class="yzm" style="width:236px;">**

**<img class="img\_captcha" src="/api/webcode/getVerifyCode" height="42" style="cursor: pointer;width:135px;" onclick="this.src ='/api/webcode/getVerifyCode?'+ Math.random();">**

**<a href="javascript:void(0);" onclick="$('.img\_captcha').click();" class="replace\_pic"></a>**

**</cite>**

10.sublime的常用配置

**{**

**"font\_size": 14,**

**"tab\_size": 2, // 设置tab键是几个空格**

**"translate\_tabs\_to\_spaces": true, // 设置tab键是几个空格**

**"update\_check": false // 禁止掉sublime总是提示更新**

**}**

11.城市的三级联动

HTML部分

<li id="shdz">

<span><i style="color:red">\*</i>收货地址：</span>

<div class="selectdiv sheng">

<select id="province" class="valid el">

<option value="0">请选择省份</option>

</select>

</div>

<div class="selectdiv shi">

<select id="country" class="valid el">

<option value="0">请选择城市</option>

</select>

</div>

<div class="selectdiv xian">

<select id="street" class="valid el">

<option value="0">区/县/街道</option>

</select>

</div>

</li>

JS部分 // 这里是调用省份的接口，及省份对应城市的接口，当然这些数据可以放在前端作为静态数据

<script>

$.ajax({

url: "/api/regionals/provinces",

success: function (data) {

var opct = "";

for (var i = 0; i < data.length; i++) {

opct += '<option value="' + data[i].id + '">' + data[i].name + '</option>';

}

$("#province").append(opct);

}

});

$("#province").live("change", function () {

var id = $(this).val();

var privateStatic = $(this).children().first().html().trim();

var children = $(this).children();

children = children.slice(1);

if (id == 0) { // 如果变回到请选择省份

$.each(children, function (i, ele) {

if (privateStatic == $(ele).html()) {

id = $(ele).val();

}

})

}

$.ajax({

url: "/api/regionals/1/" + id + "",

success: function (data) {

var opct = "";

for (var i = 0; i < data.length; i++) {

opct += '<option value="' + data[i].id + '">' + data[i].name + '</option>';

}

$("#country #street").empty();

$("#country").html('<option value="0">请选择城市</option>');

$("#street").html('<option value="0">区/县/街道</option>');

$("#country").append(opct);

}

});

})

$("#country").live("change", function () {

var id = $(this).val();

$.ajax({

url: "/api/regionals/2/" + id + "",

success: function (data) {

var opct = "";

for (var i = 0; i < data.length; i++) {

opct += '<option value="' + data[i].id + '">' + data[i].name + '</option>';

}

$("#street").empty();

$("#street").html('<option value="0">区/县/街道</option>');

$("#street").append(opct);

}

});

})

</script>

12.分页的操作

分页的插件代码

(function ($) {

var zp = {

init: function (obj, pageinit) {

return (function () {

zp.addhtml(obj, pageinit);

zp.bindEvent(obj, pageinit);

}());

},

addhtml: function (obj, pageinit) {

return (function () {

obj.empty();

if (pageinit.shownum < 5) {

pageinit.shownum = 5;

}

if (pageinit.pageNum < pageinit.shownum) {

pageinit.shownum = pageinit.pageNum;

}

var numshow = pageinit.shownum - 4;

var numbefore = parseInt((pageinit.shownum - 5) / 2);

if ((pageinit.shownum - 5) % 2 > 0) {

var numafter = numbefore + 1;

} else {

var numafter = numbefore;

}

/\*上一页\*/

if (pageinit.current > 1) {

obj.append('<a href="javascript:;" class="firstbtn">首页</a>');

obj.append('<a href="javascript:;" class="prebtn">上一页</a>');

} else {

obj.append('<a href="javascript:;" class="firstbtn">首页</a>');

obj.remove('.prevPage');

obj.append('<span class="disabled">上一页</span>');

}

/\*中间页\*/

if (pageinit.current > 4 && pageinit.pageNum > pageinit.shownum) {

obj.append('<a href="javascript:;" class="zxfPagenum">' + 1 + '</a>');

obj.append('<a href="javascript:;" class="zxfPagenum">' + 2 + '</a>');

obj.append('<span>...</span>');

}

if (pageinit.current > 4 && pageinit.current < pageinit.pageNum - numshow && pageinit.pageNum > pageinit.shownum) {

var start = pageinit.current - numbefore,

end = pageinit.current + numafter;

} else if (pageinit.current > 4 && pageinit.current >= pageinit.pageNum - numshow && pageinit.pageNum > pageinit.shownum) {

var start = pageinit.pageNum - numshow,

end = pageinit.pageNum;

} else {

if (pageinit.pageNum <= pageinit.shownum) {

var start = 1,

end = pageinit.shownum;

} else {

var start = 1,

end = pageinit.shownum - 1;

}

}

for (; start <= end; start++) {

if (start <= pageinit.pageNum && start >= 1) {

if (start == pageinit.current) {

obj.append('<span class="zxfPagenum ' + pageinit.activepage + '">' + start + '</span>');

} else if (start == pageinit.current + 1) {

obj.append('<a href="javascript:;" class="zxfPagenum ' + pageinit.activepaf + '">' + start + '</a>');

} else {

obj.append('<a href="javascript:;" class="zxfPagenum">' + start + '</a>');

}

}

}

if (end < pageinit.pageNum) {

obj.append('<span>...</span>');

}

/\*下一页\*/

if (pageinit.current >= pageinit.pageNum) {

obj.remove('.nextbtn');

obj.append('<span class="disabled">下一页</span>');

obj.append('<a href="javascript:;" class="lastbtn">尾页</a>');

} else {

obj.append('<a href="javascript:;" class="nextbtn">下一页</a>');

obj.append('<a href="javascript:;" class="lastbtn">尾页</a>');

}

/\*尾部\*/

obj.append('<span>' + '共' + '<b>' + pageinit.pageNum + '</b>' + '页，' + '</span>');

obj.append('<span>' + '到第' + '<input type="text" class="zxfinput" value="' + pageinit.current + '"/>' + '页' + '</span>');

obj.append('<span class="zxfokbtn">' + '跳转' + '</span>');

// obj.append('<span id="comfireInfo" style="color:red;display:none">（抱歉！你查找的页码不存在）</span>');

}());

},

bindEvent: function (obj, pageinit) {

return (function () {

obj.off("click");

obj.on("click", "a.prebtn", function () {

var cur = parseInt(obj.children("span.current").text());

var current = $.extend(pageinit, {

"current": cur - 1

});

zp.addhtml(obj, current);

if (typeof (pageinit.backfun) == "function") {

pageinit.backfun(current);

}

});

obj.on("click", "a.zxfPagenum", function () {

var cur = parseInt($(this).text());

var current = $.extend(pageinit, {

"current": cur

});

zp.addhtml(obj, current);

if (typeof (pageinit.backfun) == "function") {

pageinit.backfun(current);

}

});

obj.on("click", "a.nextbtn", function () {

var cur = parseInt(obj.children("span.current").text());

var current = $.extend(pageinit, {

"current": cur + 1

});

zp.addhtml(obj, current);

if (typeof (pageinit.backfun) == "function") {

pageinit.backfun(current);

}

});

// 跳转到首页

obj.on("click", "a.firstbtn", function () {

var cur = 1;

var current = $.extend(pageinit, {

"current": cur

});

zp.addhtml(obj, current);

if (typeof (pageinit.backfun) == "function") {

pageinit.backfun(current);

}

})

// 跳转到尾页

obj.on("click", "a.lastbtn", function () {

var cur = pageinit.pageNum;

var current = $.extend(pageinit, {

"current": cur

});

zp.addhtml(obj, current);

if (typeof (pageinit.backfun) == "function") {

pageinit.backfun(current);

}

})

obj.on("click", "span.zxfokbtn", function () {

var cur = parseInt($("input.zxfinput").val());

if (cur > pageinit.pageNum) {

cur = pageinit.pageNum;

console.log('pbj', $('#comfireInfo'))

$('#comfireInfo').show('normal')

layer.msg('抱歉！你查找的页码不存在。')

}

var current = $.extend(pageinit, {

"current": cur

});

var timer = setTimeout(function () {

$('#comfireInfo').hide('normal');

clearTimeout(timer)

}, 3000);

zp.addhtml(obj, current);

if (typeof (pageinit.backfun) == "function") {

pageinit.backfun(current);

}

});

// obj.on("blur", "input.zxfinput", function () {

// var cur = parseInt($("input.zxfinput").val());

// console.log('page', pageinit.pageNum)

// if (cur >= pageinit.pageNum) {

// cur = pageinit.pageNum;

// console.log('pbj',$('#comfireInfo'))

// }

// var current = $.extend(pageinit, {

// "current": cur

// });

// zp.addhtml(obj, current);

// $('#comfireInfo').show('normal')

// var timer = setTimeout(function() {

// $('#comfireInfo').hide('normal');

// clearTimeout(timer)

// }, 3000);

// if (typeof (pageinit.backfun) == "function") {

// pageinit.backfun(current);

// }

// });

obj.on("keydown", "input.zxfinput", function () {

if (event.keyCode == "13") {

$(".zxfokbtn").click();

}

});

}());

}

}

$.fn.createPage = function (options) {

var pageinit = $.extend({

pageNum: 15,

current: 1,

shownum: 9,

activepage: "current",

activepaf: "nextpage",

backfun: function () { }

}, options);

zp.init(this, pageinit);

}

}(jQuery));

DOM处的代码 <div class="n\_page"></div>

插件的使用方法：

$('.n\_page').createPage({

pageNum: totalPages, //总页码

current: 1, //当前页

backfun: function (e) {

$.ajax({

url: "/api/userReceiver/list",

dataType: "json",

contentType: "application/json",

data: {

page: e.current - 1,

},

success: function (data) {

console.log('执行代码的逻辑操作');

}

})

}

});

12.代码的两种情况比较

远程分支与分支的比较：右侧是最新分支的代码

本地与远程分支的比较：先git pull一下代码。然后右侧是本地分支的代码

13.webstorm的破解方法

第一步：打开这个路劲

c:\windows\system32\drivers\etc

第二步：用记事本打开这个路径下的hosts文件

第三步：复制“0.0.0.0 account.jetbrains.com”到hosts文件的最下方。

第四步：复制以下的内容

K71U8DBPNE--Owt3/+LdCpedvF0eQ8635yYt0+ZLtCfIHOKzSrx5hBtbKGYRPFDrdgQAK6lJjexl2emLBcUq729K1+ukY9Js0nx1NH09l9Rw4c7k9wUksLl6RWx7Hcdcma1AHolfSp79NynSMZzQQLFohNyjD+dXfXM5GYd2OTHya0zYjTNMmAJuuRsapJMP9F1z7UTpMpLMxS/JaCWdyX6qIs+funJdPF7bjzYAQBvtbz+6SANBgN36gG1B2xHhccTn6WE8vagwwSNuM70egpahcTktoHxI7uS1JGN9gKAr6nbp+8DbFz3a2wd+XoF3nSJb/d2f/6zJR8yJF8AOyb30kwg3zf5cWw==-

到话框中的Lisence Code输入框，点击OK即可

13.webstorm的破解地址

<http://idea.lanyus.com/>

14.拉取本地没有，远程有的分支

1. git fetch

1. git checkout -b 本地分支名 origin/远程分支名

15.子组件父组件的方法互调

class Son extends React.Component {

render() {

return <input onChange={this.props.onChange}/>;

}

}

class Father extends React.Component {

constructor() {

super();

this.state = {

son: ""

}

}

changeHandler(e) {

this.setState({

son: e.target.value

});

}

render() {

return <div>

<Son onChange={this.changeHandler.bind(this)}/>

<p>这里显示 Son 组件的内容：{this.state.son}</p>

</div>;

}

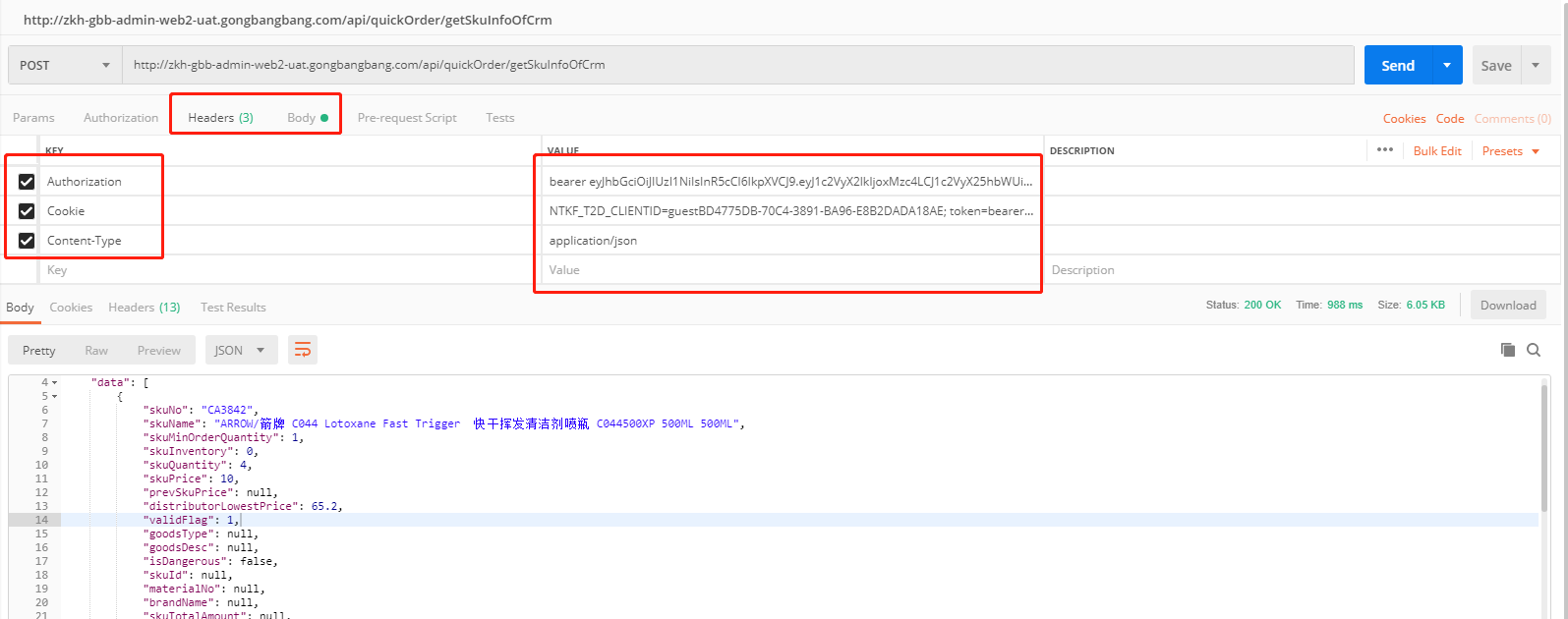
}

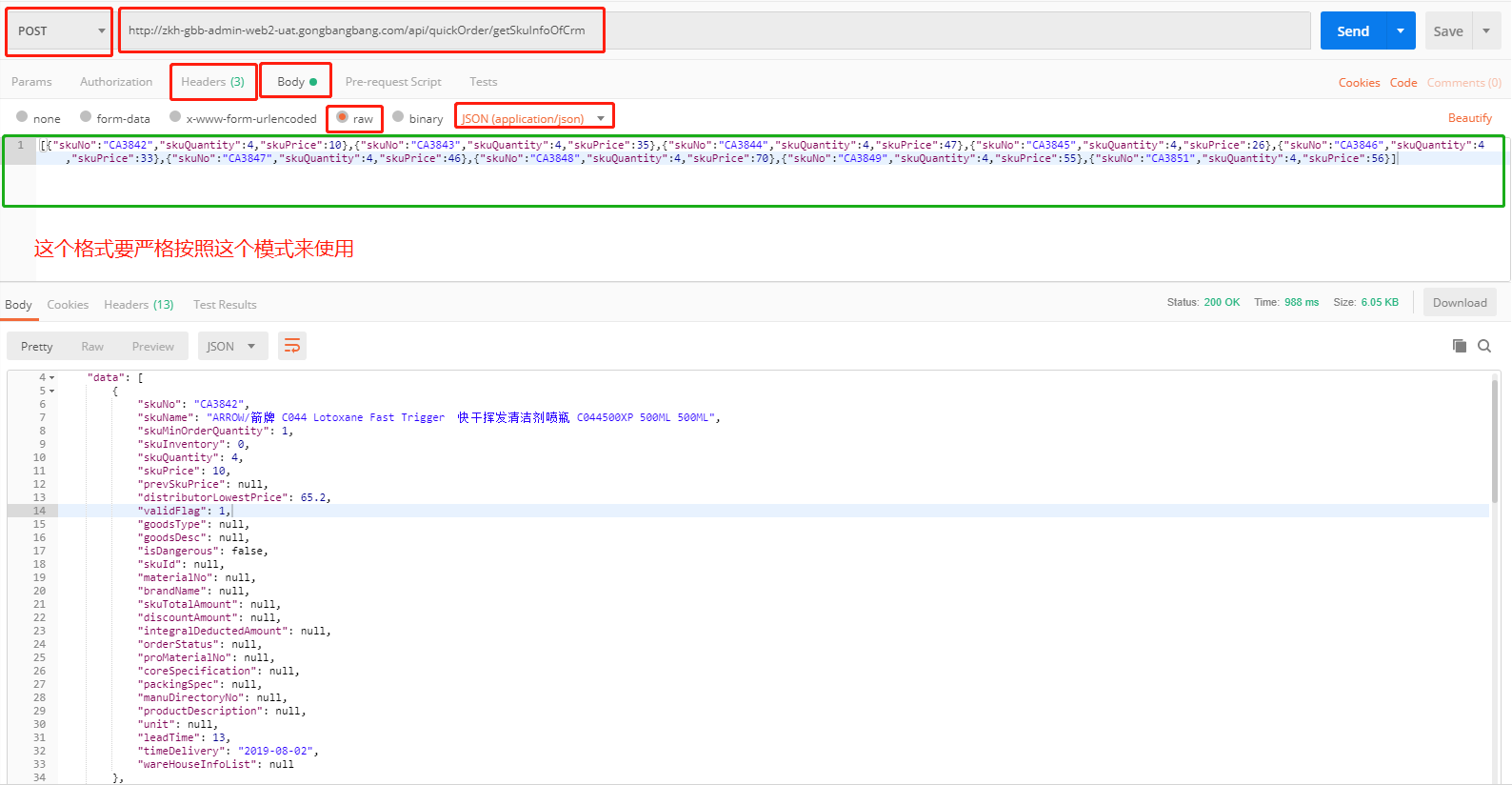
ReactDOM.render(<Father/>);

16.在dva中connect的作用

Connect的作用是将组件和models结合在一起，将models中的state绑定到组件的props中，并提供一些额外的功能，譬如:dispatch

17.postMan的使用





17.环境的问题，两步骤





18.dva中model中相关的知识点

知识点一：effects中获取model中的state的值的方法

\*importGoodsFn({ payload }, { select, call, put }) {

// 获取state中的importGoodsList,其中modelDraftDetail是namespace

const { importGoodsList } = yield select(\_=>\_.modelDraftDetail);

}

知识点二:reducer中改变state中的多层内部的某一个值时

state: {

baseInvoiceData: {

customerNumber: '',

invoiceInfoId: 0,

invoiceInfoTitle: '',

invoiceInfoType: 0,

invoiceInfoVatBank: '',

invoiceInfoVatBankAccount:'',

invoiceInfoVatRegistAddress: '',

invoiceInfoVatRegistTelephone:'',

invoiceInfoVatTaxNo:'',

userInfoId: 0,

userInfoMobile: '',

},

}

importTicketReducer(state, action) {

// 通过reducer改变state中的多层的值

return {

...state,

baseInvoiceData: {

...state.baseInvoiceData,

invoiceInfoVatBank: action.data.invoiceInfoVatBank,

invoiceInfoVatTaxNo: action.data.invoiceInfoVatTaxNo,

invoiceInfoVatRegistAddress: action.data.invoiceInfoVatRegistAddress,

invoiceInfoVatRegistTelephone:action.data.invoiceInfoVatRegistTelephone,

invoiceInfoVatBankAccount: action.data.invoiceInfoVatBankAccount,

}

};

},

知识点三：要充分结合props更新这一勾子函数

componentWillReceiveProps(nextProps) {

if (JSON.stringify(nextProps) != JSON.stringify(this.props)) {

}

}

知识点四：dispatch可以直接触发reducer中的函数，也可以先触发effect中的函数再触发reducer中的函数，effect函数可以put effect中的函数也可以触发reducer中的函数

componentWillRec

18.dva中model中相关的经验及注意事项

在使用DvaJS时，组件中的state和通过connect拿到的props分隔开，主要是通过props中的值去渲染页面，不要轻易使用componentWillReceiveProps(nextProps) {}更改组件中的state值。当需要显示客户端修改的值时，先判断客户修改了没，如果修改了用state否则用props例如：

客户修改了吗 ? this.state.Y : this.props.X

19.proxy中代理的知识点

proxy: {

'/api/\*\*': {

// target: 'https://admin.gongbangbang.com', // 线上

// target: 'zkh-gbb-admin-web2-uat.gongbangbang.com', // ms环境

target: 'http://zkh-gbb-admin-web-uat.gongbangbang.com', // 非ms环境

// target: 'http://192.168.161.93:8080',

changeOrigin: true,

},

},

注意:如果是反向代理target可以是域名的地址，也可直接是数据所在的地址(如:swagger直接调接口的地址)

20.package.json中的端口配置

默认的 "scripts": {

…,

"start": "cross-env APP\_TYPE=site umi dev",

…,

}

可修改的 "scripts": {

…,

"start": "cross-env PORT=8002 APP\_TYPE=site umi dev",

…,

}

21.快速拼接URL

const initListParam = {

enquiryNo: '', // 询价单号

statusCode: 0, // 询价状态

salesPersonId: '', // 销售id

userInfoTitle: '', // 客户名称

userInfoMobile: '', // 客户手机号

enquiryStartTime: '', // 询价开始时间

enquiryEndTime: '', // 询价结束时间

pageNum: 1, // 当前页

pageSize: 10, // 每页记录数

};

let paramsUrl = '';

for (let key in initListParam) {

paramsUrl += key + '=' + initListParam[key] + '&';

}

if (paramsUrl[paramsUrl.length - 1] == '&') {

paramsUrl = paramsUrl.slice(0, -1);

}

22.通过a标签的href一个接口去下载文件

<a className={styles.downloadApdf} href={`/api/order/downLoadOrderContractPdf/${printBarginData.orderNo}`}>下载合同</a>

23.在用Ant Design时引入图片的解决方法

Import logoBargin from '../../assets/logoBargin.png';

<img src={logoBargin} width="800" alt="工邦邦" />

23.预估工时的方法

感觉自己能多久能做完，然后记得乘以1.5 切记！切记！切记！

25.修改一个大数组中的某一个值重新赋值给model里面的state值时

dispatch({ type: 'modelFastOrder/syncFirstChangeCommon', data: { importGoodsList: [...userFnOrderNumData], }, });

26.防止重复点击事件

import { debounce } from 'lodash';

<Button type="primary" onClick={debounce(this.createDraftBargain.bind(this, 1), 500)}>

生成草稿合同

</Button>

27.想修改antDesign中的样式时

:global {

.ant-calendar-picker-clear {

display: none;

}

}

28.给一个数组中的每一项添加一个独一的key方法

import \* as shortid from 'shortid';

key: shortid.generate(),

29.import和require的区别

node编程中最重要的思想就是模块化，import和require都是被模块化所使用。

遵循规范

require 是 AMD规范引入方式

import是es6的一个语法标准，如果要兼容浏览器的话必须转化成es5的语法

调用时间

require是运行时调用，所以require理论上可以运用在代码的任何地方

import是编译时调用，所以必须放在文件开头

本质

require是赋值过程，其实require的结果就是对象、数字、字符串、函数等，再把require的结果赋值给某个变量

import是解构过程，但是目前所有的引擎都还没有实现import，我们在node中使用babel支持ES6，也仅仅是将ES6转码为ES5再执行，import语法会被转码为require

30.Ping++聚合支付系统

<script>

var payType = getUrlParam('payType');

// 定义支付渠道属性值

var channel = '';

if(payType==3){

// 支付宝

channel = 'alipay\_pc\_direct';

}else if(payType==2){

// 银联个人

channel = 'upacp\_pc'

}else if(payType==4){

// 银联公对公

channel = 'upacp\_b2b'

}else if(payType==5){

// 微信

channel = 'wx\_pub\_qr'

}

var company = "gbb";

var ordernum = getUrlParam('ordernum');

if(channel!==''){

$.ajax({

url: '/api/gateway/website/payController/PingxxPayController', // 第三方PING++聚合支付系统

type: 'post',

contentType: 'application/json;charset=utf-8',

data: JSON.stringify({

"orderId": ordernum,

"channel": channel,

"company": company,

"returnUrl": window.location.origin + "/view/shoppingCart/payNo.html?ordernum=" + getUrlParam('ordernum'),

}),

success: function (res) {

// var rsJson = JSON.parse(res)

if(res && res.success){

if(channel!== 'wx\_pub\_qr'){

// 如果不是微信支付

pingpp.createPayment(res.result, function (result, err) {

if (result == "success") {

// 只有微信公众账号 (wx\_pub)、微信小程序 (wx\_lite)、QQ 公众号 (qpay\_pub)、支付宝口碑 (alipay\_qr)

// 支付成功的结果会在这里返回，其他的支付结果都会跳转到 extra 中对应的 URL。

} else if (result == "fail") {

// data 不正确或者微信公众账号/微信小程序/QQ 公众号/支付宝口碑支付失败时会在此处返回

} else if (result == "cancel") {

// 微信公众账号、微信小程序、QQ 公众号、支付宝口碑支付取消支付

}

});

}else{

// 如果是通过微信支付

// console.log('如果yes是微信支付');

// var url = window.location.origin + "/api/gateway/website/tools/qrcode?text=" + res.result.credential['wx\_pub\_qr']

// window.open(url)

}

}else{

layer.msg("该订单已支付!");

}

},

error: function (res) {

console.log('PING++聚合支付系统支付失败', res)

}

})

}

</script>

31.iframe如何把外部的页面嵌入到自己的页面中

<div>

<iframe

style={{ width: '100%', height: 800, border: 0 }}

src="https://das.base.shuju.aliyun.com/token3rd/dashboard/view/pc.htm?pageId=a1cca13e-b79b-4dcd-b98c-f5a0a8cc6c80&accessToken=ad6618b532ed09a626653ba47e89361f"

/>

</div>

31.Dva.js中设置成同步的问题

yield put({

type: 'effectsCouponsList',

payload: {

req: payload.req.activityId,

},

});

yield take('effectsCouponsList/@@end');

32.MobX方法的一些理解

@inject 在模块内用 @inject('Store')，将 Store 注入到 props 上,保证结构的一致性

使用 @observer ，将组件变为观察者，观察状态的变化，当状态变化时，组件也会做相应的更新。

33.递归的简单理解

var Ajson = {

name: '数据13',

children: [

{

name: '数据131',

},

{

name: '数据132',

}

]

}

var arr = [];

function common(json) {

if (!json.children) {

arr.push(json);

} else {

for(var i = 0; i < json.children.length; i++) {

common(json.children[i]);

}

}

return arr;

}

console.log(common(Ajson));

34.package.json和package-lock.json的区别

package.json 这个文件是 npm init 时创建的一个文件，会记录当前整个项目中的一些基础信息。而 package-lock.json 这个文件却是 node\_modules 文件夹或者 package.json 文件发生变化时自动生成的。Package-lock.json主要功能是确定当前安装包的依赖，以便后续重新安装的时候，生成相同的依赖，忽略项目开发过程中有些依赖已经发生的更新。

35.监听浏览器左上角的上一步下一步的事件

// popstate

window.addEventListener('popstate', (res) => {

  console.log('res是什么呢', res.path[0].location.pathname);

})

36.更改路由，不刷新页面

window.history.pushState({}, '', 'http://localhost:5000/');

37.监听ctrl+enter事件

keypress(e) {

      // 键盘的回车事件

      if (e.which === 13 && e.ctrlKey) {

        this.textareaSearch();

      }

}

38.自适应

const colWidth = {

   xs: {span: 24},

   sm: {span: 24},

   lg: {span: 8},

   xl: {span: 8},

   xxl: {span: 6}

}

<Row>

<Col {…colWidth}>哈喽啊</Col>

</Row>

39.递归将后端返回的一个数组树形数据结构的字段换成自己需要用到的字段：

var catalogArr= JSON.parse(" [{"catalogId":-1,"parentId":0,"level":1,"catalogName":"未分类","children":null,"inoperable":true,"canAddChildren":false,"hasChildren":false,"remark":"系统预设分类，不可编辑和删除"},{"catalogId":45,"parentId":0,"level":1,"catalogName":"一级分类","children":[{"catalogId":46,"parentId":45,"level":2,"catalogName":"二级分类","children":[{"catalogId":47,"parentId":46,"level":3,"catalogName":"三级分类","children":[{"catalogId":48,"parentId":47,"level":4,"catalogName":"四级分类","children":null,"inoperable":false,"canAddChildren":false,"hasChildren":false,"remark":null}],"inoperable":false,"canAddChildren":true,"hasChildren":true,"remark":null}],"inoperable":false,"canAddChildren":true,"hasChildren":true,"remark":null}],"inoperable":false,"canAddChildren":true,"hasChildren":true,"remark":null},{"catalogId":49,"parentId":0,"level":1,"catalogName":"A分类","children":[{"catalogId":50,"parentId":49,"level":2,"catalogName":"A-1分类","children":[{"catalogId":51,"parentId":50,"level":3,"catalogName":"A-1-1分类","children":[{"catalogId":52,"parentId":51,"level":4,"catalogName":"A-1-1-1","children":null,"inoperable":false,"canAddChildren":false,"hasChildren":false,"remark":null}],"inoperable":false,"canAddChildren":true,"hasChildren":true,"remark":null}],"inoperable":false,"canAddChildren":true,"hasChildren":true,"remark":null}],"inoperable":false,"canAddChildren":true,"hasChildren":true,"remark":null}]");

// 定义一个方法

let getTree = (jsonParam) => {

jsonParam.value = jsonParam.catalogId; // 因为我需要用到的value值跟返回的catalogId是对应的，因此直接给jsonParam添加了一个value属性，并用catalogId给value赋值

jsonParam.title = jsonParam.catalogName; // 因为我需要的值跟返回的code是对应的，就直接给jsonParam添加了一个value属性 并把code 给它赋值

// jsonParam.children = jsonParam.lowerLeves; // 因为我要用到的key也不对(如：无children)，所以这个lowerLeves我也换成了children

// 如果孩子存在

if (jsonParam.children && jsonParam.children.length > 0) {

for(let i = 0; i < jsonParam.children.length; i++) {

// 因为树形结构 不确定内部层次到底有几层 所以要在循环里边做回调

// 在回调之前为了严谨起见，要先判断一下这个里边还有没有下级

if(jsonParam.children[i]) {

getTree(jsonParam.children[i])

}

}

}

return jsonParam

}

var newArr = [];

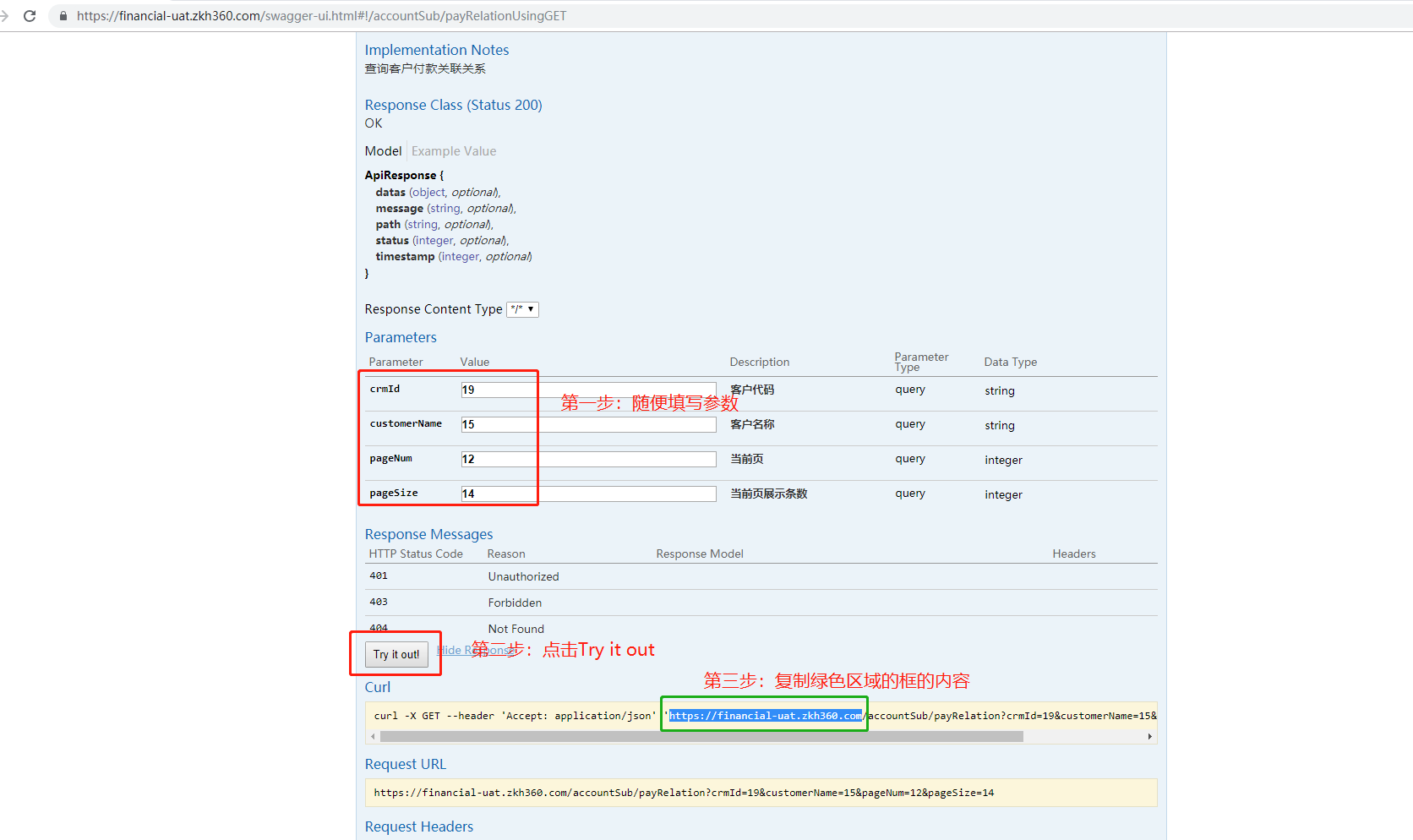
catalogArr.forEach((item, index) => {

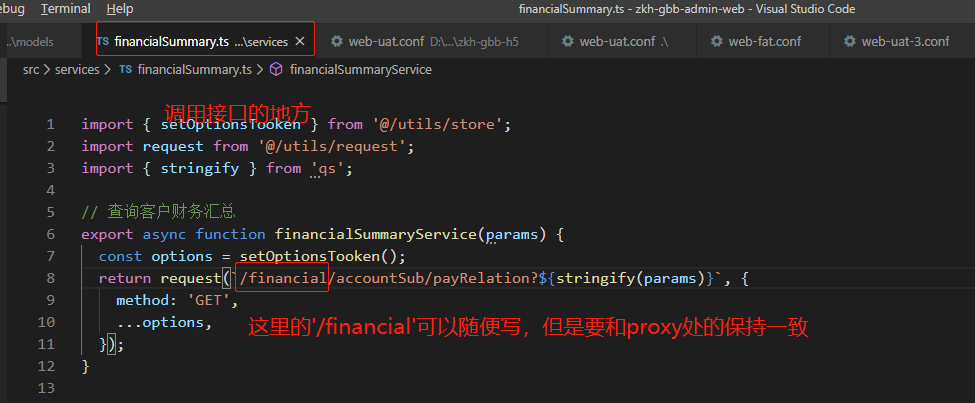
newArr.push(getTree(item));

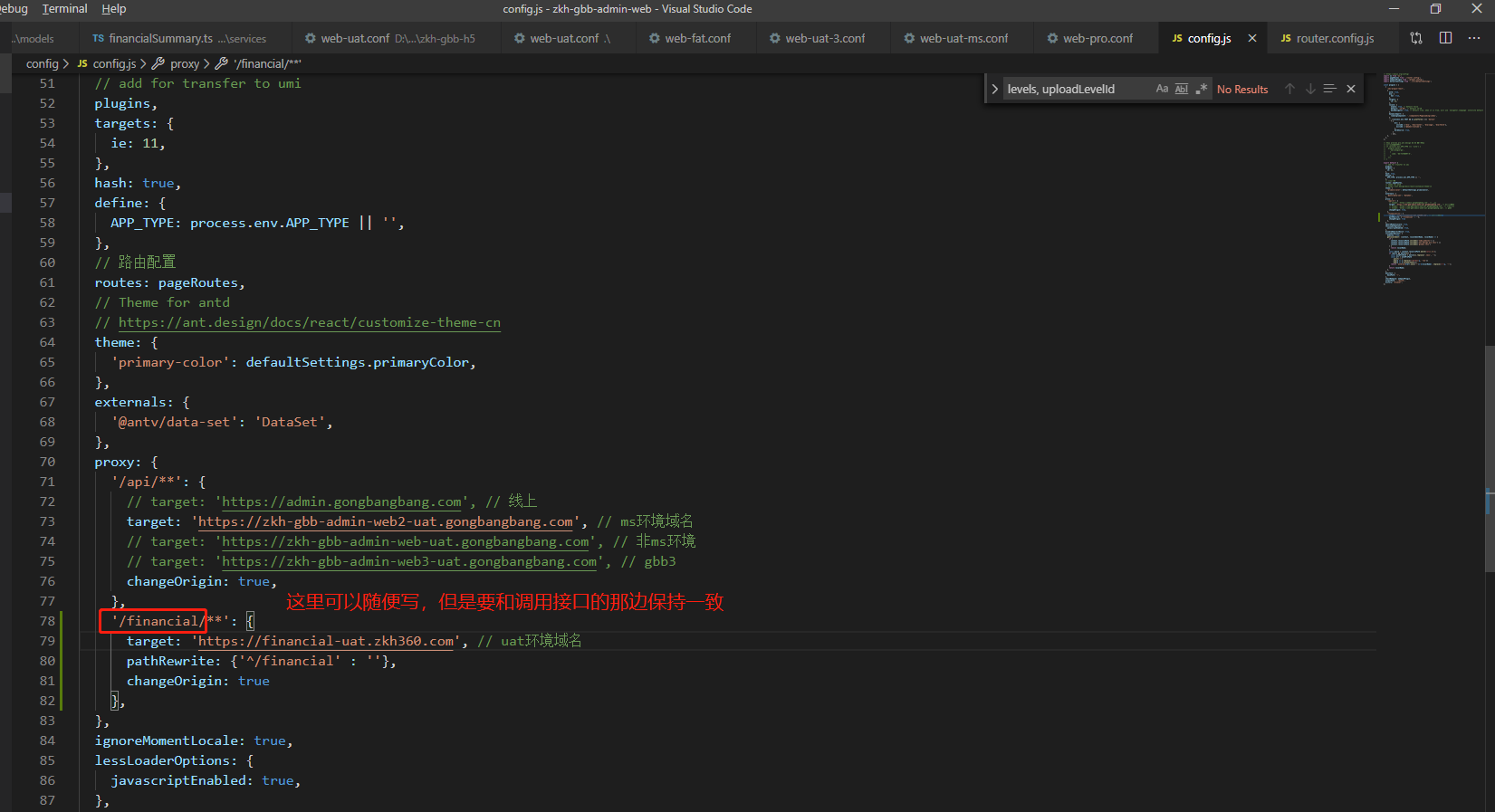
})

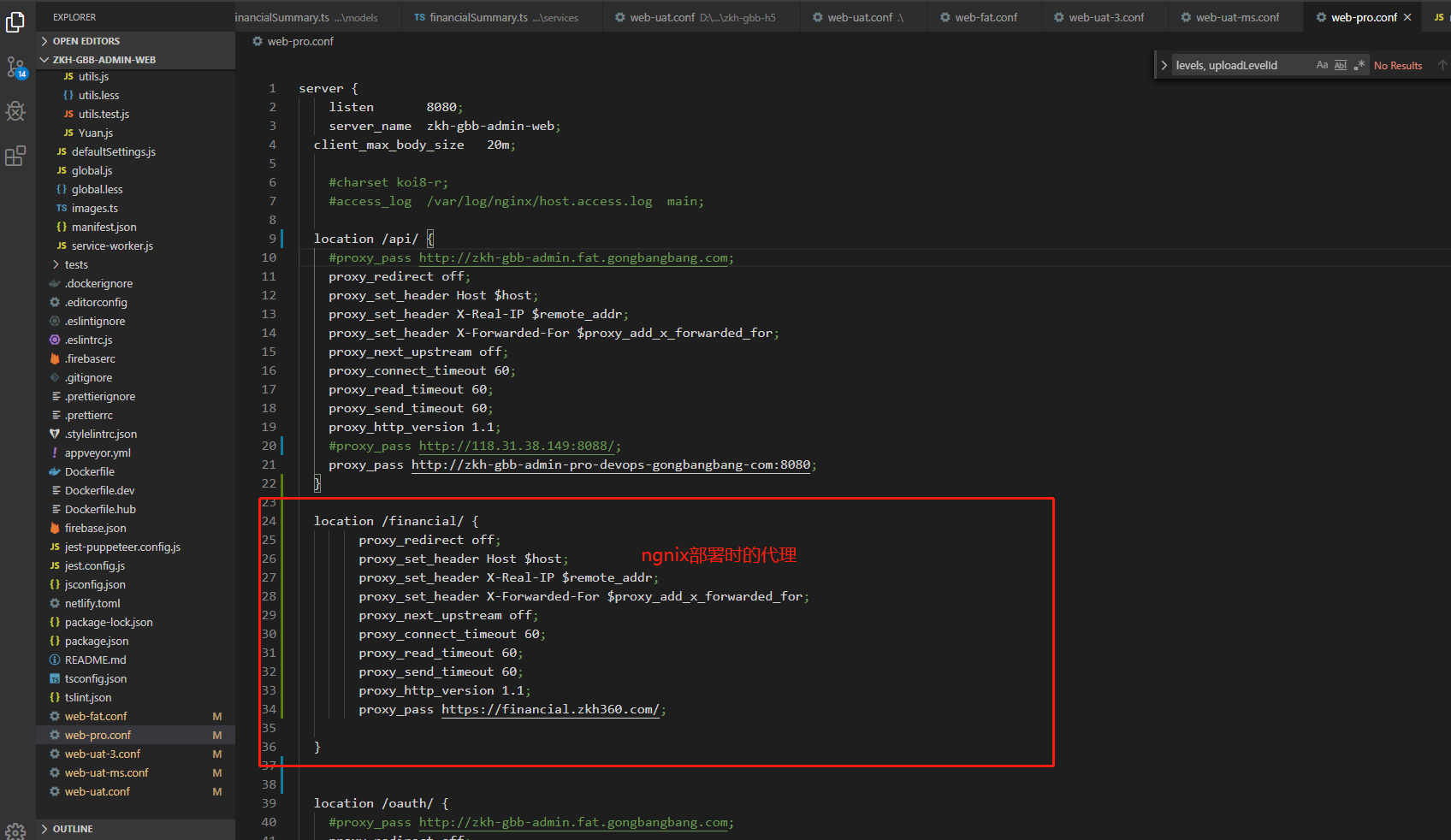
console.log('newArr ==== ', newArr);

40.如何结合后端给的swagger配置proxy代理









41.彻底清除浏览器缓存，不在让用户清除缓存

<meta http-equiv="cache-control" content="no-cache,no-store, must-revalidate" />

<meta http-equiv="pragma" content="no-cache" />

<meta http-equiv="Expires" content="0" />

location ~ .\*\.(htm|html)?$ {

#原来这样设置的不管用

#expires -1;

#现在改为，清除缓存

add\_header Cache-Control "private, no-store, no-cache, must-revalidate, proxy-revalidate";

access\_log on;

}

42.导出文件的方法

方法一：a标签法<a href={`/api/product/stock/export?${stringify(newSearchParam)}`}>批量导出</a>

方法二：按钮法

let request = new XMLHttpRequest();

    request.open('GET', `/api/product/stock/export?${stringify(newSearchParam)}`);

    request.setRequestHeader('Authorization', Cookies.get(COOKIE\_KEYS.TOKEN));

    request.responseType = 'arraybuffer';

    request.onload = function(e) {

      var file = new Blob([this.response], { type: 'application/vnd.ms-excel' });

      let elink = document.createElement('a');

      elink.download = '商品备货清单';

      elink.style.display = 'none';

      elink.href = URL.createObjectURL(file);

      document.body.appendChild(elink);

      elink.click();

      document.body.removeChild(elink);

    };

    request.send();

43.将数组里某个属性相同的对象合并成一个数组（数组中包含数组）

var array=[

{id:1,name:'小明',sex:'男'},

{id:2,name:'小刚',sex:'男'},

{id:3,name:'小红',sex:'女'},

{id:4,name:'小花',sex:'女'},

{id:5,name:'小甜甜',sex:'女'},

]

var obj={},newArr=[];

array.forEach(function(item,suffix){

//根据对象的属性是唯一的，将值作为对象的属性名

if(!obj[item.sex]){

var arr=[];

arr.push(item);

newArr.push(arr);

obj[item.sex]=item;

}else{

newArr.forEach(function(value,index){

//如果已经存在 就循环新组的值将值插入属性相同的数组里 为了防止重复添加 只要和第一个比较就可以了

if(value[0].sex==item.sex){

value.push(item)

}

})

}

})

console.log(obj);

console.log(newArr)

44.package.json当严格限制了代码风格时可去掉如下红色代码即可，将代码提上去。

{

"name": "fanchisee-web",

"version": "1.0.0",

"private": true,

"description": "加盟前台",

"scripts": {

"analyze": "cross-env ANALYZE=1 umi build",

"build": "node --max\_old\_space\_size=4096 node\_modules/umi/bin/umi build",

"deploy": "npm run site && npm run gh-pages",

"dev": "npm run start:dev",

"fetch:blocks": "pro fetch-blocks --branch=umi@3 && npm run prettier",

"gh-pages": "cp CNAME ./dist/ && gh-pages -d dist",

"i18n-remove": "pro i18n-remove --locale=zh-CN --write",

"lint": "umi g tmp && npm run lint:js && npm run lint:style && npm run lint:prettier",

"lint-staged": "lint-staged",

"lint-staged:js": "eslint --ext .js,.jsx,.ts,.tsx ",

"lint:fix": "eslint --fix --cache --ext .js,.jsx,.ts,.tsx --format=pretty ./src && npm run lint:style",

"lint:js": "eslint --cache --ext .js,.jsx,.ts,.tsx --format=pretty ./src",

"lint:prettier": "prettier --check \"\*\*/\*\" --end-of-line auto",

"lint:style": "stylelint --fix \"src/\*\*/\*.less\" --syntax less",

"prettier": "prettier -c --write \"\*\*/\*\"",

"start": "umi dev",

"start:dev": "cross-env REACT\_APP\_ENV=dev MOCK=none umi dev",

"start:no-mock": "cross-env MOCK=none umi dev",

"start:no-ui": "cross-env UMI\_UI=none umi dev",

"start:pre": "cross-env REACT\_APP\_ENV=pre umi dev",

"start:test": "cross-env REACT\_APP\_ENV=test MOCK=none umi dev",

"test": "umi test",

"test:all": "node ./tests/run-tests.js",

"test:component": "umi test ./src/components",

"tsc": "tsc"

},

"husky": {

"hooks": {

"pre-commit": "npm run lint-staged"

}

},

"lint-staged": {

"\*\*/\*.less": "stylelint --syntax less",

"\*\*/\*.{js,jsx,ts,tsx}": "npm run lint-staged:js",

"\*\*/\*.{js,jsx,tsx,ts,less,md,json}": [

"prettier --write"

]

},

"browserslist": [

"> 1%",

"last 2 versions",

"not ie <= 10"

],

"dependencies": {

"@ant-design/icons": "^4.0.0",

"@ant-design/pro-layout": "^5.0.0",

"@ant-design/pro-table": "^2.0.0",

"antd": "^4.0.0",

"classnames": "^2.2.6",

"echarts": "^4.7.0",

"echarts-for-react": "^2.0.15-beta.1",

"lodash": "^4.17.11",

"moment": "^2.24.0",

"omit.js": "^1.0.2",

"path-to-regexp": "2.4.0",

"qs": "^6.9.0",

"react": "^16.8.6",

"react-dom": "^16.8.6",

"react-helmet-async": "^1.0.4",

"react-pdf": "^4.1.0",

"shortid": "^2.2.15",

"umi": "^3.0.0",

"umi-request": "^1.0.8",

"use-merge-value": "^1.0.1"

},

"devDependencies": {

"@ant-design/pro-cli": "^1.0.18",

"@types/classnames": "^2.2.7",

"@types/express": "^4.17.0",

"@types/history": "^4.7.2",

"@types/jest": "^25.1.0",

"@types/lodash": "^4.14.144",

"@types/qs": "^6.5.3",

"@types/react": "^16.9.17",

"@types/react-dom": "^16.8.4",

"@types/react-helmet": "^5.0.13",

"@types/shortid": "0.0.29",

"@umijs/fabric": "^2.0.5",

"@umijs/plugin-blocks": "^2.0.5",

"@umijs/preset-ant-design-pro": "^1.0.1",

"@umijs/preset-react": "^1.3.0",

"@umijs/preset-ui": "^2.0.9",

"chalk": "^3.0.0",

"cross-env": "^7.0.0",

"cross-port-killer": "^1.1.1",

"enzyme": "^3.11.0",

"express": "^4.17.1",

"gh-pages": "^2.0.1",

"husky": "^4.0.7",

"jsdom-global": "^3.0.2",

"lint-staged": "^10.0.0",

"mockjs": "^1.0.1-beta3",

"prettier": "^2.0.1",

"pro-download": "1.0.1",

"stylelint": "^13.0.0"

},

"optionalDependencies": {

"puppeteer": "^2.0.0"

},

"engines": {

"node": ">=10.0.0"

},

"checkFiles": [

"src/\*\*/\*.js\*",

"src/\*\*/\*.ts\*",

"src/\*\*/\*.less",

"config/\*\*/\*.js\*",

"scripts/\*\*/\*.js"

]

}

# 45．用第三方脚手架打包文件过大的问题，Nginx 如何 开启gzip 来提高页面加载速度。

gzip on;

gzip\_min\_length 1k;

gzip\_buffers 4 16k;

#gzip\_http\_version 1.0;

gzip\_comp\_level 2;

gzip\_types text/plain application/javascript application/x-javascript text/css application/xml text/javascript application/x-httpd-php image/jpeg image/gif image/png;

gzip\_vary off;

gzip\_disable "MSIE [1-6]\.";

// 解释说明：

第1行：开启Gzip

第2行：不压缩临界值，大于1K的才压缩，一般不用改

第3行：buffer，就是，嗯，算了不解释了，不用改

第4行：用了反向代理的话，末端通信是HTTP/1.0，有需求的应该也不用看我这科普文了；有这句的话注释了就行了，默认是HTTP/1.1

第5行：压缩级别，1-10，数字越大压缩的越好，时间也越长，看心情随便改吧

第6行：进行压缩的文件类型，缺啥补啥就行了，JavaScript有两种写法，最好都写上吧，总有人抱怨js文件没有压缩，其实多写一种格式application/javascript 就行了

第7行：跟Squid等缓存服务有关，on的话会在Header里增加"Vary: Accept-Encoding"，我不需要这玩意，自己对照情况看着办吧

第8行：IE6对Gzip不怎么友好，不给它Gzip了

# 46．Iframe的高度自适应方法。

<template>

  <div style="height:auto;">

    <iframe style="width:100%;height:100%;" :src="bdTokenUrl" frameborder="0" scrolling="auto" id="bdIframe"></iframe>

  </div>

</template>

<script type="text/javascript">

  // 放大缩小刷新页面

  var normalWidth = window.innerWidth;

  var normalHeight = window.innerHeight;

  window.onresize = function(){

    if(normalWidth>window.innerWidth || normalHeight < window.innerHeight){

      location.reload();

    }

  }

  export default {

    name: 'IndividualBoard',

    data(){

      return{

        bdTokenUrl : ''

      }

    },

    created() {

      this.getUrl();

    },

    mounted(){

      /\*\*

       \* iframe-宽高自适应显示

      \*/

      this.$nextTick(()=>{

        const oIframe = document.getElementById('bdIframe');

        const deviceHeight = document.documentElement.clientHeight;

        let headerHeight = document.getElementsByClassName('nav\_header')[0].offsetHeight; // 这个高度没用的原因，怀疑是

        console.log('deviceHeight++', deviceHeight);

        console.log('headerHeight===', headerHeight);

        oIframe.style.height = (Number(deviceHeight) - 87 - 20) + 'px'; // 数字是空白的padding

      });

    },

    methods: {

      getUrl() {

        let empCode = sessionStorage.getItem('empCode');

        this.bdTokenUrl = `http://alca.deppon.com/deppon/p1587876675422\_Auto\_6page.zul?empCode=${empCode}`

      },

    }

  }

</script>

<style lang="less" scoped>

</style>

# 47．获取数组树形结构中某一个key值对应的整个item

var allTree = [

  // 所有的DOM节点

  {

    path: '/portraitModule/contractPortrait',

    name: 'portraitModule',

    nameDesc: '画像模型',

    permission: true,

    component: null,

    button: null,

    icon: '',

    type: 'link',

  },

  {

    path: '/systemManagement',

    name: 'systemManagement',

    nameDesc: '系统管理',

    permission: true,

    component: null,

    button: null,

    icon: '',

    type: 'module',

    children: [

      {

        path: '/logManagement',

        name: 'logManagement',

        nameDesc: '日志管理',

        permission: false,

        component: null,

        button: null,

        icon: 'el-icon-document',

        type: 'link',

        children: [

          {

            path: '/logManagement/systemLog',

            name: 'systemLog',

            nameDesc: '系统日志',

            permission: false,

            component: null,

            button: null,

            icon: '',

            type: 'link',

            children: null

          },

          {

            path: '/logManagement/loginLog', // 这一层才是配置路由的地址

            name: 'loginLog',

            nameDesc: '登录日志',

            permission: false,

            component: null,

            button: null,

            icon: '',

            type: 'link',

            children: null

          },

          {

            path: '/logManagementisitLog',

            name: 'visitLog',

            nameDesc: '访问日志',

            permission: false,

            component: null,

            button: null,

            icon: '',

            type: 'link',

            children: null

          },

          {

            path: '/logManagementisitStatistical',

            name: 'visitStatistical',

            nameDesc: '访问统计',

            permission: false,

            component: null,

            button: null,

            icon: '',

            type: 'link',

            children: null

          }

        ]

      },

    ]

  },

]

var returnedItem

var recursiveTree = function(arr, key){

  for(let i=0;i<arr.length;i++){

      if( arr[i].name===key ){

        returnedItem = arr[i]

        return

      }else if(arr[i].children && arr[i].children.length > 0){

        recursiveTree(arr[i].children, key);  //递归调用

      }

  }

}

 console.log(recursiveTree(allTree, 'systemManagement'))

# 48．通过beforeEach给页面加权限

注意事项：

1.beforeEach必须放在main.js中

2.beforeEach里面必须只有两种next操作，防止死循环

例如：

router.beforeEach((to, from, next) => {

  let returnedItem

  let recursiveTree = function(arr, key){

    for(let i=0;i<arr.length;i++){

      if( arr[i].name===key ){

        returnedItem = arr[i]

        return

      }else if(arr[i].children && arr[i].children.length > 0){

        recursiveTree(arr[i].children, key);  //递归调用

      }

    }

  }

  const allTree = store.state.allTree

  recursiveTree(allTree, to.name)

  if (!localStorage.getItem('empCode') || (returnedItem && !returnedItem.permission)) {

    next('/')

  }else{

    next()

  }

})

# 49．[vue项目中请求头为applicationx-www-form-urlencoded的参数传递](https://www.cnblogs.com/thinkguo/p/11277313.html)

1. const params = new URLSearchParams();

params.append('programId', row.programId)

2.然后再将该对象传递到接口参数

program\_submit(params).then(res => {

if (res.data.code == '0') {

that.$openMessage('提交成功', 'success');

this.getAllprogram();

} else {

that.$openMessage(res.data.msg, 'error');

}

}).catch(e => {

that.$openMessage(e, 'error');

})

},

3.对应的axios的接口如下，需要加个headers

export function program\_submit(params) {

return axios.put(servers + '/program/submit', params, {

headers: { 'Content-Type': 'application/x-www-form-urlencoded' }

});

}

如果axios.create配置了则配置封装函数

const service = axios.create({

  baseURL: process.env.NODE\_ENV === 'development' ? '/api/alca-service' : '/alca-service',

  headers: {

    post: {

      'Content-Type': 'application/x-www-form-urlencoded'

    }

  }

})

export const form = function (url, params) {

  return service.post(url, params, {

    headers: { 'Content-Type': 'application/x-www-form-urlencoded'}

  })

}