111assign 合并对象并具有相同属性的对象

var o1 = { a: 1, b: 1, c: 1 };

var o2 = { b: 2, c: 2 };

var o3 = { c: 3 };

var obj = Object.assign({}, o1, o2, o3);

console.log(obj); // { a: 1, b: 2, c: 3 }

注释

*/\*\*  
 \** ***@method*** *超出行显示省略号  
 \** ***@param*** *{string} contextClass 超出行显示省略号  
 \** ***@param*** *{number} [lineNum=2] 第几行需要省略号  
 \** ***@param*** *{number} [lineHeight=28] 行高  
 \** ***@param*** *{number} [safeNum=30] 用于提高dom操作性能，每行安全的字符数，如30，2行，不会检查30\*2以内的字符是否需要填充‘...’  
 \** ***@return*** *null 返回值说明  
 \*/*

多行省略号

*/\*\*  
 \** ***@method*** *超出行显示省略号,在update中调（会将样式自动设置成 height:auto）;  
 \** ***@param*** *{string} contextClass 超出行显示省略号  
 \** ***@param*** *{number} [lineNum=2] 第几行需要省略号  
 \** ***@param*** *{number} [lineHeight=28] 行高  
 \** ***@param*** *{number} [safeNum=30] 用于提高dom操作性能，每行安全的字符数，如30，2行，不会检查30\*2以内的字符是否需要填充‘...’  
 \** ***@return*** *null 返回值说明  
 \*/*dot: **function** (contextClass, lineNum, lineHeight, safeNum) {  
 lineNum = lineNum || 2  
 lineHeight = lineHeight || 28  
 safeNum = lineHeight || 30  
 **let** el = document.getElementsByClassName(contextClass)  
 // console.log(el);  
 **for** (**var** j = 0; j < el.length; j++) {  
 **let** wN = 0 // 字数，图片算一个字  
 el[j].style.height = 'auto'  
 el[j].style.width = 'auto'  
 el[j].style.maxHeight = 'none'  
 // console.log('j', j, 'el[j].offsetHeight', el[j].offsetHeight)  
 **var** pT = parseInt(window.getComputedStyle(el[j]).paddingTop)  
 **var** pB = parseInt(window.getComputedStyle(el[j]).paddingBottom)  
 // console.log('pT', pT, 'pB', pB, lineNum \* lineHeight + pT + pB)  
 **if** (el[j].offsetHeight < lineNum \* lineHeight + pT + pB) {  
 // console.log('< 25', el[j].offsetHeight)  
 **continue** } **else** {  
 // console.log('> 25')  
 // break  
 }  
 **let** str = el[j].innerHTML  
 // console.log(str);  
 **for** (**var** i = 0; i < str.length; i++) {  
 **var** a // 是否是img的标志位  
 **var** resNum  
 **var** res  
 **if** (str[i] === '<' && str.slice(i + 1, i + 5) === 'img ') {  
 // i = i+1  
 res = str.slice(i)  
 // console.log('res', res)  
 resNum = res.indexOf('>')  
 // console.log(resNum.);  
 a = 1  
 i = i + resNum  
 wN++  
 } **else** {  
 wN++  
 }  
 // console.log('str.substr(0, i)', str.substr(0, i + 1))  
 **if** (wN > 2 \* safeNum) {  
 el[j].innerHTML = str.substr(0, i + 1) + '...'  
 // console.log('i=', i, 'el[j].offsetHeight', el[j].offsetHeight) // , 'str', str  
 **if** (el[j].offsetHeight > lineNum \* lineHeight + pT + pB) {  
 // el[j].style.overflow = 'hidden';  
 **if** (a === 1) {  
 a = 0  
 **if** (j === 0) {  
 // console.log('strLast', str.substr(0, i - resNum))  
 }  
 el[j].innerHTML = str.substr(0, i - resNum) + '...'  
 // console.log(t.substr(0, i - resNum) + '....');  
 } **else** {  
 a = 0  
 // console.log('strLast', str.substr(0, i - 1))  
 // el[j].innerHTML = t.substr(0, i);  
 el[j].innerHTML = str.substr(0, i - 1) + '...'  
 }  
 **break** } **else** {  
 a = 0  
 }  
 el[j].innerHTML = str  
 }  
 }  
 }  
}

1111样式

111max

## 可能的值

|  |  |
| --- | --- |
| **值** | **描述** |
| none | 默认。定义对元素被允许的最大高度没有限制。 |
| *length* | 定义元素的最大高度值。 |
| *%* | 定义基于包含它的块级对象的百分比最大高度。 |
| inherit | 规定应该从父元素继承 max-height 属性的值。 |

对象的浅拷贝

function easyClone(Obj) {

var objNew = {};

for ( var i in Obj) {

objNew[i] = Obj[i];

}

return objNew;

其实就是将每个原对象的属性和值复制到新对象上去，当然我们也可以使用Object.assign() 方法可以把任意多个的源对象自身的可枚举属性拷贝给目标对象，然后返回目标对象，同时Object.assign() 也是浅拷贝。

浅拷贝因为没有递归循环检查对象的每个值是否是对象，而是直接进行了赋值，所以如果某个值是对象的时候就会出现问题，所以在一般情况下我们需要用深拷贝来进行备份。

对象的深拷贝

最简单的深拷贝

var b = JSON.parse(JSON.stringify(a))

但是这种深拷贝有一定的局限性，第一：无法复制函数，第二：原形链没了，对象就是object，所属的类没了。

其实简单的深拷贝只需要我们递归调用浅拷贝就可以了：

function deepCopy(obj) {

　　var objNew = objNew || {};

　　for (var i in obj) {

　　　　if (typeof p[i] === 'object') {

　　　　　　objNew[i] = (p[i].constructor === Array) ? [] : {};

　　　　　　deepCopy(obj[i], objNew[i]);

　　　　} else {

　　　　　　　objNew[i] = obj[i];

　　　　}

　　}

　　return objNew;

}

111遍历对象

for(var key in obj){

//只遍历对象自身的属性，而不包含继承于原型链上的属性。

if (obj.hasOwnProperty(key) === true){

keys.push(key);

values.push(obj[key]);

}

}

alert("keys is ："+keys+" and values is ："+values);

111遍历数组

var arr = [1,2,3,4];

arr.forEach(function(value,index,array){

    array[index] == value;    //结果为true

    sum+=value;

    });

console.log(sum);    //结果为 10

111 file

<input type="file" placeholder="" class="col-sm-12" v-on:change="fileReady($event)">

fileReady: **function** (e) {  
 **this**.fileState = **true  
 var** file = e.target.files[0]  
 console.log(e.target)  
},

**var** form = **new** FormData()  
**var** params2 = {  
 couponId: **this**.couponIdModal  
}  
 // form.append("couponId", this.couponIdModal)  
 form.append("excelFilePath", **this**.file)

$.ajax({  
 url:URL+ '/loanBk/coupon/importUsers?'+str,  
 type:'post',  
 // dataType:'json',  
 // contentType:'multipart/form-data; boundary=----WebKitFormBoundarylZiVIC4IH7OVaGwq',  
 headers: pubZ.headers,  
 data: params,  
 processData:**false**, //这个一定要  
 contentType: **false**, //这个一定要  
 success: **function** (result) {  
 **if**(result.code=='200000'){

processData:**false**, //这个一定要  
 contentType: **false**, //这个一定要

异步多ajax，次数不定，jquery ，eval 拼字符 when then

**var** str1 = '$.when('  
 **var** strs = ''  
 **var** str3 = ').then(function(a,b) {\n' +  
 ' var count = 0\n' +  
' console.log(arguments);\n' +  
 ' for(var i = 0; i<arguments.length; i++){\n' +  
 ' console.log("arguments[i]", arguments[i], arguments[i]);\n' +  
 ' console.log("arguments[i][0].code", arguments[i][0].code, arguments[i][0].data);\n' +  
  
 'if(arguments[i][0].code == "200000" && arguments[i][0].data == 1) {\n' +  
 'count++\n' +  
 ' console.log("陈工一个", count);\n' +  
  
 '}\n' +  
'};\n' +  
'if(count == arguments.length){ShowWarningAlert("警告","全部成功")}else{ShowWarningAlert("警告","未全部成功")}\n' +  
'\n' +  
 ' })'  
  
 **var** \_this = **this  
 var** userCouponIdList = []  
 **this**.tableData.forEach(**function** (value) {  
 **if** (value.check == **true**) {  
 /\*userCouponIdList.push(value.userCouponId)  
 strs = strs + 'api\_coupon\_revokeUserCoupon(' + '{userCouponId:'+value.userCouponId+'}' + ', function (result) {\n' +  
  
 ' }),'\*/  
 userCouponIdList.push(value.userCouponId)  
 strs = strs + 'api\_coupon\_revokeUserCoupon(' + '{userCouponId:'+value.userCouponId+'}' + '),'  
}  
 })  
 **var** strLast = str1 + strs +str3  
 console.log('strLast', strLast)  
 **if** (userCouponIdList.length > 0) {  
 eval(strLast)  
 }

api\_coupon\_revokeUserCoupon 方法中一定要返回 jquery AJAX对象

**function** api\_coupon\_revokeUserCoupon(params){  
 **return** $.ajax({  
 url:URL+ '/loanBk/coupon/revokeUserCoupon',  
 type:'patch',  
 dataType:'json',  
 // contentType:'text',  
 headers: pubZ.headers,  
 data: params,  
 });  
}

$.when( $.ajax("test.aspx") ).then(function(ajaxArgs){

alert(ajaxArgs[1]); /\* ajaxArgs is [ "success", statusText, jqXHR ] \*/

});

ajaxArgs是一个数组，是所有请求的详情