1.函数内部变量调用外部函数，

  
  
错误：var panduan = pandua(jieguo)  
  
2.JS new出来的构造函数属于一个对象没有返回值。

如果var c = a1(4);改为var c = new a1(4);则弹出object  
<body>

<button onclick="a2()">666</button>

<script type="text/javascript">

var b=0;

function a1(b){

b=b+1;

return b;

};

function a2(){

var c = a1(4);

alert(c);

}

</script>

</body>

3.async异步函数

varasyncReadFile= async function(){

var f1 = await readFile('/etc/fstab');

var f2 = await readFile('/etc/shells');

console.log(f1.toString());

console.log(f2.toString());};

async 表示函数里有异步操作，await 表示紧跟在后面的表达式需要等待结果

<http://www.ruanyifeng.com/blog/2015/05/async.html>  
  
需求场景:

函数A里面调用函数B,拿到B的返回值继续往下走,函数B为异步函数(比如百度地图接口)

async A(){

console.log(‘bbefore’)

let cc = await B

console.log(‘b after’)

}

B(){

return new Promise(resolve){

….ajax

if(success){ resolve (data)}

else { reject(‘xxx’) }

}

}

必须等到cc拿到结果,才能往下执行console.log

await后面应该跟一个promise对象

比如 const b = await 123

会被转成const b = await promise.reject(123)



4.ES6  
 (1)一个类必须有constructor方法，如果没有显式定义，一个空的constructor方法会被默认添加.  
 constructor方法默认返回实例对象（即this），完全可以指定返回另外一个对象。  
  
 (2) function add(...values) {  
  
 ES6引入rest参数（形式为“...变量名”），用于获取函数的多余参数，这样就不需要使用arguments对象了。rest参数搭配的变量是一个数组，该变量将多余的参数放入数组中。  
<http://www.cnblogs.com/huansky/p/5684020.html>  
  
rest参数之后不能再有其他参数（即只能是最后一个参数），否则会报错。

// 报错function f(a, ...b, c) {

// ...

}

[http://es6.ruanyifeng.com/#docs/function#rest](http://es6.ruanyifeng.com/#docs/function)参数  
  
 (3)  
类名(组件名)一定要用大写开头，否则自定义的组件无法编译，识别不出来。

类中定义render函数要注意两点，见代码注释。

render() {//开头花括号一定要和小括号隔一个空格，否则识别不出来

return <ol>//标签前一半一定要和return一行

{

React.Children.map(this.props.children, function (child) {

return <li>{child}</li>;

})

}

</ol>;

}

在class中使用class的变量或者方法，一定要加个this。如this.handlerclick。

es6 绑定事件需要onClick={this.func1.bind(this)}。

这样func1和bind里面的参数‘this’的作用域才绑定到了一起（注意es5是不需要这个bind(this)的），func1中如果有this.name这类语句，相当于是使用参数‘this’里面的变量值;或者使用箭头函数func1= (e)=> {函数体}

5.foreach,map,reduce区别

区别:

foreach不能中断,for in只能循环对象,for of都可以  
实际应用：

把"10,-6.1|28,-6.0" 中的10，28组成数组["10","28"]

-6.1,-6.0组成另一个数组["-6.1","-6.0"]



函数中用filter的时候 需要在数组前面加个return来返回值

6.es6 引用赋值  
 var a = {foo:'bar'};

var b = a;

a.foo = 'good';

console.log(b == a)//true  
  
 对象和数组的赋值是引用赋值， a 和 b 只是一个引用符号，其所指向的对象实体不同（比如 a -> object#001, b -> object#002），js的对象（数组）相等判断是根据是否指向同一个对象实体来的确定的

7.constructor  
 constructor(props){

super(props)

}  
这段话是用来创建this的，this由super（）创建的，子组件加这段代码能得到父组件传过来的props  
  
8.JS函数只执行一次  
 var isFirst = true;

function doS(){

if(isFirst){

doSth...//这里写你要做什么

isFirst = false;

}

}

9.处理字符串 split()  
<http://www.w3school.com.cn/jsref/jsref_split.asp>  
"2:3:4:5".split(":") //将返回["2", "3", "4", "5"]

"|a|b|c".split("|") //将返回["", "a", "b", "c"]

10.查询字符串  
 使用substring（）方法  
 倒叙查询方法：<http://www.cnblogs.com/guilongliang/p/5527790.html>

例子：JSON对象属性名以UOD结尾的且值为1改为2，值为-1改为3 该怎么做  


Map+substring+for in  
  
  


11.负数小数点进两位  
(-0.014 \* 100).toFixed(1)

直接乘100不行

12.两个页面之间跳转传参数  
<http://www.cnblogs.com/zhu-xingyu/p/5577118.html>  
  
第一个页面  
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>direct1.html</title>

</head>

<body>

<div><a id="a1" href="direct2.html">点击此处</div>

<script>

var a1=document.getElementById("a1");

var person={

store\_id:"nk11",//模拟,上线改为从参数获得

date:201730,//模拟,上线改为从参数获得

version:20170810

};

a1.href=a1.href+"?"+person.store\_id+'|'+person.date+'|'+person.version;

</script>

</body>

</html>  
  
第二个页面  
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>direct2.html</title>

</head>

<body>

<script>

var getInfo=window.location.search.slice(window.location.search.lastIndexOf("?")+1);

var cc = getInfo.split("|")

console.log(getInfo);

</script>

</body>

</html>

13.合并单元格

SpanGrid(tabObj\_f, cellindex, beginRow) { *//合并单元格, id ,第几列, 开始行*

let colIndex=cellindex;

let rowBeginIndex=beginRow;

let tabObj=document.getElementById(tabObj\_f)

**if** (tabObj!= null) {

let i, j;

let intSpan;

let strTemp='';

**for** (i=rowBeginIndex; i< (tabObj.rows.length-rowBeginIndex); i++) {

intSpan=1;

strTemp=tabObj.rows[i].cells[colIndex].innerText;

**for** (j =i+1; j <tabObj.rows.length; j++) {

**if** (strTemp.trim() ==tabObj.rows[j].cells[colIndex].innerText.trim() ) {

intSpan++;

tabObj.rows[i].cells[colIndex].rowSpan=intSpan;

tabObj.rows[j].cells[colIndex].style.display="none";

}

**else** {

tabObj.rows[i].cells[colIndex].style.display="table-cell"

tabObj.rows[j].cells[colIndex].style.display="table-cell"

i= j -1

**break**;

}

}

}

}

},

14.JS 获得第几周的星期日期

<http://bbs.csdn.net/topics/392030605>

ISO标准算法  
function getDate(period,week){

        //period:第几周，week:星期几

        if(/^([1-4]?\d|5[12])$/.test(period) && /^[0-6]$/.test(week)){

            var ymd = new Date(new Date().getFullYear(),0,1);

            ymd.setDate(1 + period \* 7 - ymd.getDay() + week);

            return ymd.getFullYear() +"-"+ (ymd.getMonth() + 1) +"-"+ ymd.getDate();

        }else{

            return "参数错误";

        }

    }

console.log(getDate(0,0));//获取第一周星期一日期

console.log(getDate(3,3));//获取第四周星期四日期

console.log(getDate(10,6));//获取第十一周星期天日期

补充:判断当前日期为第几周 <http://momentjs.cn/docs/#/get-set/iso-week-year/>

或者  
getYearWeek(dateString){ //时间转周数

var da =dateString;//日期格式2015-12-30

//当前日期

var date1 = new Date(da.substring(0,4), parseInt(da.substring(5,7)) - 1, da.substring(8,10));

//1月1号

var date2 = new Date(da.substring(0,4), 0, 1);

//获取1月1号星期（以周一为第一天，0周一~6周日）

var dateWeekNum=date2.getDay()-1;

if(dateWeekNum<0){dateWeekNum=6;}

if(dateWeekNum<4){

//前移日期

date2.setDate(date2.getDate()-dateWeekNum);

}else{

//后移日期

date2.setDate(date2.getDate()+7-dateWeekNum);

}

var d = Math.round((date1.valueOf() - date2.valueOf()) / 86400000);

if(d<0){

var date3 = (date1.getFullYear()-1)+"-12-31";

return this.getYearWeek(date3);

}else{

//得到年数周数

var year=date1.getFullYear();

var week=Math.ceil((d+1 )/ 7);

return year+"-"+week;

}

},



https://blog.csdn.net/lxz547236/article/details/50433917

15.获取URL参数  
GetRequest() {

var url = location.search;

var theRequest = new Object();

if (url.indexOf("?") != -1) {

var str = url.substr(1);

var strs = str.split("&");

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

theRequest[strs[i].split("=")[0]]=unescape(strs[i].split("=")[1]);

}

}

return theRequest;

}

16.数组倒叙输出  
需求:api倒叙排列 data:reverse()

17.JS 对象里调用函数结果当属性值  


18.用for循环生成对象时,外面要包层if 避免每次引用这个对象都计算一遍  


19.JS深复制

因为JS数组和对象的直接赋值,修改的是引用.  
所以如果数组里嵌套对象,对象里又嵌套数组,很可能操作的时候会出问题,解决办法是data =JSON.parse(JSON.stringify(data))

这个方法当有属性是undefined或null的时候会被移除掉  
<https://www.zhihu.com/question/23031215>

20.alert函数去掉服务器地址显示

http://blog.csdn.net/u010883006/article/details/77985034



21.断点  
在代码中空一行写bugger即可或者debugger

22.error  
<http://www.jb51.net/article/14481.htm>

打印http里的error

error.response

Property:   
  name: 错误名   
  number: 错误号   
  message: 错误信息,多同description   
  description: 描述   
 FF Only 属性   
  fileName: 错误发生的文件   
  stack: 错误发生时的调用堆栈

23.MD5  
参数为0的时候有时候需要传字符串0(根据后端)

24.判断类型  
Object.prototype.toString.call(data) == "undefined"

不能用typeof因为type数组和null的结果都是object

25.两个div同步滚动

//编辑器和预览区同步滚动

let preview = document.querySelector('#preview');

let editor = document.querySelector('#editor');

preview.addEventListener('scroll', (e)=> {

e.stopPropagation();

editor.scrollTop = preview.scrollTop;

});

editor.addEventListener('scroll', (e)=> {

e.stopPropagation();

preview.scrollTop = editor.scrollTop;

});

26.遍历tree 找出叶子节点的层级

function getArray(data,name)

{

for (var i in data) {

console.log('i',i);

console.log('datai',data[i].children);

if (data[i].name == name) {

console.log(data[i]);

break;

} else {

getArray(data[i].children, name);

}

}

}

<http://blog.csdn.net/yf275908654/article/details/52215321>

27.在光标出插入内容

<input id=d1 type=text onclick="test('123',this)" />

<input id=d2 type=text onclick="test('123',this)" />

<script>

function test(str,obj){ //在光标处插入内容

var tc = obj//document.getElementById(id);

var tclen = tc.value.length;

tc.focus();

if(typeofdocument.selection != "undefined")

{

document.selection.createRange().text = str;

}

else

{

tc.value = tc.value.substr(0,tc.selectionStart)+str+tc.value.substring(tc.selectionStart,tclen);

}

}

</script>

<http://bbs.csdn.net/topics/380193980/>

28.JS中变量赋值为markdown

不能有<script>‘’’要转义成’/’/’/’



29.生成二维码

<http://www.cnblogs.com/zxyun/p/5682779.html>

30.自定义属性

标签里写 data-\*

js读的时候读object.dataset对象中定义的属性 不带\*

<https://www.cnblogs.com/dolphinX/p/3348458.html>

31.数字千分位分隔,金钱逗号分隔

var a = 222122122.6754;

var b = a.toFixed(2).replace(/(\d)(?=(\d{3})+\.)/g, '$1,');

32.阻止冒泡



33.ajax向后端传递数组

当后端语言为PHP直接传递即可,当为Python时候,需要加一个JSON.stringify包裹住数组

34.获取最近7天

get\_last7day(){ //获取最近7天日期并渲染

var myDate = new Date();

myDate.setDate(myDate.getDate() - 7);

var dateArray = [];

var dateTemp;

var flag = 1;

for (var i = 0; i< 7; i++) {

dateTemp = myDate.getFullYear()+"-"+(myDate.getMonth()+1)+"-"+myDate.getDate();

dateArray.push(dateTemp);

myDate.setDate(myDate.getDate() + flag);

}

this.times[0] = dateArray[0]

this.times[1] = dateArray[6]

this.times = this.times.map( (item,index)=>{ //月份 日 补成双数

let f2 = ''

let f3 = ''

if( item.split('-')[1] < 10 ){

f2 = '0'+item.split('-')[1]

}else{

f2 = item.split('-')[1]

}

if( item.split('-')[2] < 10 ){

f3 = '0'+item.split('-')[2]

}else{

f3 = item.split('-')[2]

}

return item.split('-')[0] +'-'+ f2+'-'+f3

})

},

35.call和apply

区别:call参数需要按顺序传,apply参数需要放在数组里传

用法:模块化编程的时候改变this的上下文

比如a对象有个方法a1和变量a2,方法a1会返回a2,

var a ={

a2:7,

a1:function(){ return this.a2 }

}

var b = a.a1

b()

结果:undefined  
解析:b函数的上下文是window  
 var a ={

a2:7,

a1:function(){ return this.a2 }

}

var b = a.a1

b.call(a)  
结果:7

解析:call改变了b函数执行时的上下文为a

另一个例子:

猫吃鱼，狗吃肉.

有天狗想吃鱼了

猫.吃鱼.call(狗，鱼)

<http://blog.csdn.net/lovecarpenter/article/details/54089490>

<https://www.zhihu.com/question/20289071>

36.Object.defineProperty

试着运行下面函数

test\_1(){

var Book = {}

var name = ''

Object.defineProperty(Book,'name',{

set:(value)=>{

name = value;

console.log('取的书名为'+value)

},

get:()=>{

return '<<'+name+'>>'

}

})

Book.name = 'vue大全'

console.log(Book.name)

console.log(Book)

},

37.判断两个时间日期不能超过30天

已知 times:[‘2018-02-03’,’2018-03-07’]

var oneTime = new Date().setTime(new Date(this.times[0]).getTime());

var twoTime = new Date().setTime(new Date(this.times[1]).getTime());

if( oneTime + 3600 \* 1000 \* 24 \* 30 <twoTime){

this.alert\_status = true

this.alert\_title = "提示"

this.alert\_text = '时间范围不能超过31天,请修改后再试'

return

}

38.JS 实现new  
function mock\_new(data){

 return  function (){

   let obj = {}

   obj.\_proto\_ = data.prototype

data.apply(obj,arguments) //obj代替函数里的this

   return obj

   }

}

function Person(name,age){

this.name = name;

this.age = age;

}

var p2 = mock\_new(Person)('jack',25)

p2.name

<http://web.jobbole.com/91017/>

new的简单应用

function Animal(name){

this.name = name;

}

Animal.color = "black";

Animal.prototype.say = function(){

console.log("I'm " + this.name);

};

var cat = new Animal("cat");

console.log(cat.color) //undefined

console.log(cat.name) //cat

console.log(cat.say()) //I,m cat

console.log(Animal.name) //Animal 返回的是函数内置属性name,内置属性name默认为函数名(仅函数对象)

https://www.cnblogs.com/AaronNotes/p/6529492.html

39.JS 数组对象 按照对象属性增删数组  
需求:数组如果没有某个对象就新增,有就删除  
使用Array.prototype.findIndex,根据对象的关键属性值进行判断

findIndex()方法返回数组中满足提供的测试函数的第一个元素的索引。否则返回-1

比如给定一个数组对象某个属性值,找出这个对象的数组位置





40.数组转字符串  
  
<https://www.liaoxuefeng.com/wiki/001434446689867b27157e896e74d51a89c25cc8b43bdb3000/001435119854495d29b9b3d7028477a96ed74db95032675000>

41.按照数组索引方式获取对象值  
let arr = Object.keys(obj)  
data[ arr[i] ]

42.终止调用函数,子函数中断父函数

在被调用函数出加上throw new Error即可!

比如ajax参数验证

或者子函数返回false ,父函数判断子函数结果,子函数结果正确要返回 true,例子:

function A(){  
 if(xxxx){return false}  
return true

}

function B(){

if(!A())return

}

43.检验数组

校验是否整数:Number.isInteger(data)

校验小数,位数保留两位不足补0:

先input设置只能输入数字,再:

Number(data).toFixed(2)

44.dirname:当前目录

45.promise

promise一般和fetch搭配,手写promise用处不大

promise类似关键字class,一个函数把原来由return返回的结果改成:成功由resolve返回,错误由reject返回.后面再用then调用成功后执行的函数,用catch调用失败后执行的函数



46.命令行创建文件格式

touch test.js

47.解构赋值

作用1:可以选择性获得函数返回的值

{

**function** f(){

**return** [1,2,3,4,5]

}

**let**a,b;

[a,,,b]=f();

console.log(a,b)

}

作用2:如果函数返回一个数组,获取数组的第一个值,剩下的填入另一个数组

{

**function** f(){

**return** [1,2,3,4,5]

}

**let**a,b,c,d;

[c,...d]=f()

console.log(c,d)

}

作用3:提取json

{

**let**metaData={

title:'abc',

test:[{

title:'test',

desc:'description'

}]

}

**let** {title:esTitle,test:[{title:cnTitle}]}=metaData

console.log(esTitle,cnTitle)

}

作用4 输入模块指定方法

const {SourceMapConsumer,SourceNode}= require("source-map");

作用5函数传参

functionsetCookie(name,value,options){

options = options || {};

let secure = options.secure,

path = options.path,

domain = options.domain;

}

setCookie("type","js",{

secure:true,

path:'http://baidu.com'

})

function setCookie(name,value,{secure,path,domain}){

     //设置cookie的代码

}

setCookie("type","js",{

   secure:true,

   path:'http://baidu.com'

})

<https://www.cnblogs.com/tangjiao/p/9019671.html>

48.export

es6中export不能直接输出值,输出的是值得引用,比如export 1会报错

export default 导出对应的是import from es6规范

(node不支持ES6的import)

module.exports 导出对应的是 require 只node支持

https://segmentfault.com/a/1190000010426778

49.模板字符串

传统方式单引号里面跟字符串的时候,如果换行需要拆开两个字符串并用+号连接,直接换行编译器会报警告,比如



ES6语法 使用反引号``可以直接换行

50.await

加上await感觉像写同步代码



51.判断两个字符串是否相等

https://www.cnblogs.com/liaokaichang/p/7567433.html

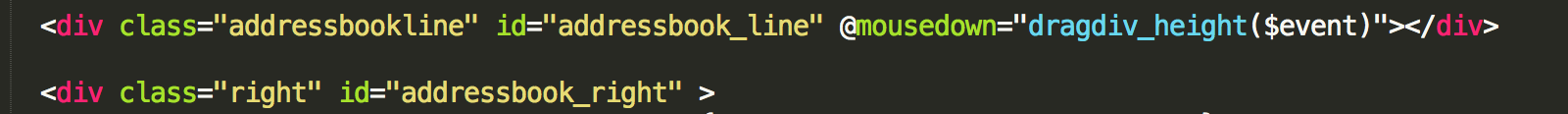
1. trim() == b.trim()

52.判断变量 true 或者 false

var a = 100

console.log(!!a)

53.拖拽div边框





54.回调函数

forEach( (item,index)=>{console.log(item)} )

(item,index)=>{console.log(item)}这个匿名函数就是回调函数,被在函数中使用的函数是回调函数

55.input输入长度处理,中文6个字符12个

拿到input输入的值,watch然后调用下面函数



56.谷歌浏览器的console为异步,再修改对象返回对象的时候可以先深度遍历给一个新对象,返回新对象

56.JS 事件捕获和冒泡

在addEventListener中事件默认在冒泡阶段执行,但如果是在子组件中监听事件可能会监听不到,比如vue监听子组件滚动事件,这时候需要把addEventListener的第三参数改为true,即在捕获阶段执行事件即可,

然后在触发的函数中判断event.bubbles的值就能拿到是冒泡还是捕获,vue中在子组件监听,子组件滚动是捕获父组件滚动是冒泡

57.curl

ssh命令 可以在终端起到postman效果

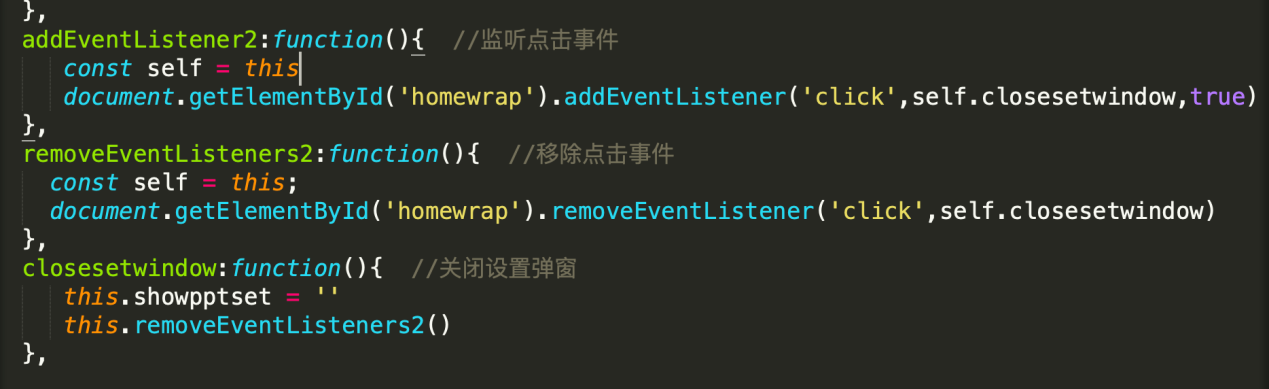
比如 curl –d ‘name=lilei&age=23’<http://www.baidu.com>  
向百度发送post请求,参数name:lilei age:23

58.indexeddb

Indexeddb在返回promise的时候要close连接,否则升级数据库的时候会报pending

59.JS弹窗

无遮罩层的弹窗 ,打开之后在最外层DOM加个click事件监听函数,第三参数为true表示捕获阶段,点弹窗外执行移除函数 弹窗内执行回调函数,回调函数里再包移除函数



60.时间格式化

functiondateFormat(fmt, date) {

let ret;

const opt = {

"Y+": date.getFullYear().toString(), // 年

"m+": (date.getMonth() +1).toString(), // 月

"d+": date.getDate().toString(), // 日

"H+": date.getHours().toString(), // 时

"M+": date.getMinutes().toString(), // 分

"S+": date.getSeconds().toString() // 秒

// 有其他格式化字符需求可以继续添加，必须转化成字符串

};

for (let k in opt) {

ret =newRegExp("("+ k +")").exec(fmt);

if (ret) {

fmt=fmt.replace(ret[1], (ret[1].length ==1) ? (opt[k]) : (opt[k].padStart(ret[1].length, "0")))

};

};

returnfmt;

}

let date =newDate()

dateFormat("YYYY-mm-dd HH:MM", date)

61.三个选项判断



62.生成一定范围内N个随机数

newNumber(start,end){

return Math.round(Math.random()\*(end-start)+start);//生成在[start,end]范围内的随机数值，只支持不小于0的合法范围

},

isHaveThisNumber(para,num){

//拒绝无效的数组

if(typeof(para) == "object")

{

if(para.length==0)

{

console.log("数组为空");

return false;

}

}

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

if(para[i]==num){

console.log("与第【"+(i+1)+"】个元素重复，正在重新生成...");

return true;//与目标数组有重复

}

}

return false;

},

newRandomNumbersWithNoRepeat(start,end,size){

var para=new Array();//目标随机数组

var rnum;//当前随机数

var currentIndex=0;//当前随机数组的索引

if(start>end||start<0||end<0||size<0){

console.log("无效参数");

return;

}

if(end-start+1<size){//验证随机数个数是否超出随机数范围

console.log("随机数范围大于随机区间");

return;

}

for(var i=0;i<size;i++){//生成 size 个不重复的随机数

rnum=this.newNumber(start,end);//获取随机数

if(this.isHaveThisNumber(para,rnum)){//是否已经存在

while(isHaveThisNumber(para,rnum)){//获取新的随机数 直到不重复

rnum=this.newNumber(start,end);//重新获取随机数

console.log("发现新的随机数："+rnum);

}

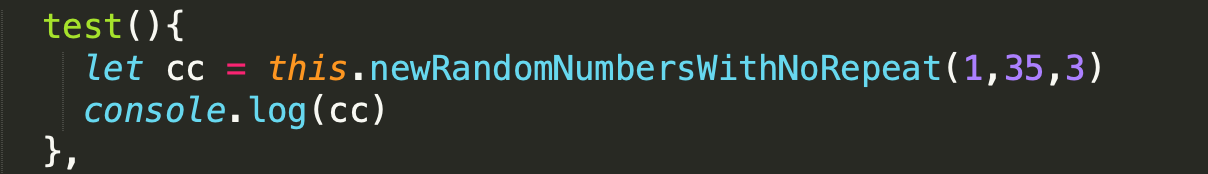
}

para[currentIndex++]=rnum;//添加到现有数字集合中

}

return para;

}



<https://blog.csdn.net/brucethl/article/details/79123929?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-24.nonecase&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-24.nonecase>

63.监听手势

GetSlideAngle(*dx*, *dy*) {

        return Math.atan2(dy, dx) \* 180 / Math.PI;

    },

    //根据起点和终点返回方向 1：向上，2：向下，3：向左，4：向右,0：未滑动

    GetSlideDirection(*startX*, *startY*, *endX*, *endY*) {

*const* \_that = this

*let* dy = endY - startY;

*let* dx = endX - startX;

*let* result = 0;

        //如果滑动距离太短

        if(Math.abs(dx) < 2 && Math.abs(dy) < 2) {

            return result;

        }

*let* angle = \_that.GetSlideAngle(dx, dy);

        if(angle >= -45 && angle < 45) {

            result = 4;

        }else if (angle >= 45 && angle < 135) {

            result = 1;

        }else if (angle >= -135 && angle < -45) {

            result = 2;

        }

        else if ((angle >= 135 && angle <= 180) || (angle >= -180 && angle < -135)) {

            result = 3;

        }

        return result;

    },

    bengin\_touch(){

*let* startX, startY;

*const* \_that = this

          document.addEventListener('touchstart',*function* (*ev*) {

              startX = ev.touches[0].pageX;

              startY = ev.touches[0].pageY;

          }, false);

          document.addEventListener('touchend',*function* (*ev*) {

*var* endX, endY;

              endX = ev.changedTouches[0].pageX;

              endY = ev.changedTouches[0].pageY;

*var* direction = \_that.GetSlideDirection(startX, startY, endX, endY);

              switch(direction) {

                  case 0:

                      alert("没滑动");

                      break;

                  case 1:

                      alert("向上");

                      break;

                  case 2:

                      alert("向下");

                      break;

                  case 3:

                      alert("向左");

                      break;

                  case 4:

                      alert("向右");

                      break;

                  default:

              }

          }, false);

    },

64.拓展运算符

如果用户自定义的属性，放在扩展运算符后面，则扩展运算符内部的同名属性会被覆盖掉。

let aWithOverrides = { ...a, x: 1, y: 2 };

// 等同于

let aWithOverrides = { ...a, ...{ x: 1, y: 2 } };

// 等同于

let x = 1, y = 2, aWithOverrides = { ...a, x, y };

// 等同于

let aWithOverrides = Object.assign({}, a, { x: 1, y: 2 });

如果把自定义属性放在扩展运算符前面，就变成了设置新对象的默认属性值。

let aWithDefaults = { x: 1, y: 2, ...a };

// 等同于

let aWithDefaults = Object.assign({}, { x: 1, y: 2 }, a);

// 等同于

let aWithDefaults = Object.assign({ x: 1, y: 2 }, a);

65.链判断运算符 ES2020

const firstName = message?.body?.user?.firstName || 'default';

const fooValue = myForm.querySelector('input[name=foo]')?.value