/\*

鼠标坐标

\*/

document.onclick = *function* (*event*) {

//获取page坐标三步法 在页面位置等于 = 看得见的+看不见的

*var* event = event || window.event;

*var* pagex = event.pageX || event.clientX + scroll().left;

*var* pagey = event.pageY || event.clientY + scroll().top;

*console*.log(pagex +" "+ pagey);

}

//=========================================================================

*const* utils = {

/\*

element.scrollHeight // 返回整个元素的高度（包括带滚动条的隐蔽的地方）

element.scrollWidth // 返回整个元素的宽度（包括带滚动条的隐蔽的地方）offsetWidth不包括

1.内容超出盒子，显示内容的宽/高

2.没有超出，火狐、谷歌、IE8+ 显示盒子的宽/高，

3.IE67不管内容有没有盒子，都显示内容宽/高

// ==============================

element.scrollTop和element.scrollLeft // 网页被浏览器卷去的头部和左边部分,有兼容性问题

1.未声明 DTD（谷歌）: document.body.scrollTop （获取body元素）

2.已经声明DTD（火狐，IE）: document.documentElement.scrollTop （获取html元素）

3.火狐/谷歌/ie9+以上 : window.pageYOffset （获取页面Y轴）

兼容写法:为了兼容，不管有没有 DTD，封装自己的scrollTop

window.onscroll = function () {

scrollTop()

}

\*/

scrollTop: *function*() {

return window.pageYOffset || document.documentElement.scrollTop || document.body.scrollTop || 0; // 都不兼容初始值为0

// 或

// return = document.documentElement.scrollTop + document.body.scrollTop; // 这两个值总会有一个恒为0

},

/\*

\* 获取页面被卷起的top、left

\* window.onscroll = function () {

\* var t = scroll().top;

\* var l = scroll().left;

\* document.title = t+" "+l;

\* }

\* 判断页面有没有DTD约束：document.compatMode === "BackCompat" 未声明

\* document.compatMode === "CSS1Compat" 已经声明

\* \*/

scroll: *function*() {

//浏览器如果支持window.pageYOffset;那么直接使用

//不能单写 window.pageYOffset 因为支持的话 但为0，就不执行了

if(window.pageYOffset != undefined) {

return {

top: window.pageYOffset,

left: window.pageXOffset

};

}else if(document.compatMode === "BackCompat") { //浏览器未声明DTD约束

return {

top: document.body.scrollTop,

left: document.body.scrollLeft

};

}

return { // 以上都不支持直接返回

top: document.documentElement.scrollTop,

left: document.documentElement.scrollLeft

};

},

/\*

盒子调用：指盒子本身宽高

浏览器调用：可视区域大小

document.body.clientHeight = width + padding

document.body.clientWidth = height + padding

element.clientWidth; // 获取网页可视区域宽度（两种用法）---不包括边框，边距或滚动条

element.clientHeight; // 获取网页可视区域高度（两种用法）---不包括边框，边距或滚动条

clientWidth与clientX 区别

event.clientX // 鼠标距离可视区域左侧距离（event调用）

event.clientY // 鼠标距离可视区域上侧距离（event调用）

\*/

/\*

\* 获取当前可视区域(浏览器)的width、height

\* 调用

\* window.onresize = function () {

\* document.title = client().width+" "+client().height;

\* }

\* \*/

client: *function*() {

/\*return {

width: window.innerWidth || document.documentElement.clientWidth || document.body.clientWidth,

height: window.innerHeight || document.documentElement.clientHeight || document.body.clientHeight,

}

或

\*/

if(window.innerWidth != undefined) {

return { // ie9及其以上的版本

width: window.innerWidth,

height: window.innerHeight

}

}else if(document.compatMode === "CSS1Compat") {

return { // 标准模式(有DTD)("CSS1Compat") IE

"width": document.documentElement.clientWidth,

"height": document.documentElement.clientHeight

}

}

return { // 怪异模式 （没有DTD）

width: document.body.clientWidth,

height: document.body.clientHeight

}

},

/\*

获得屏幕宽高(显示器的分辨率), 指显示器所能显示的像素有多少

\*/

screen: *function*() {

return {

width: window.screen.width,

height:window.screen.height

}

},

/\*

封装className,兼容IE678

var boxArr = fn("box");

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

// className注意

// boxArr[i].className = "current"; // 只添加了current类名

//boxArr[i].className = "box current"; // 等价于

boxArr[i].className += " current"; // 在原来的类名追加current

}

\*/

className: *function*(*str*) {

// 1.元素有一个class

/\* if(document.getElementsByClassName){

return document.getElementsByClassName(str);

}

// 不支持

var arrAllElements = document.getElementsByTagName("\*");

var arr = [];

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

if(arrAllElements[i].className == str){

arr.push(arrAllElements[i]);

}

}

return arr;\*/

// 2.元素有多个class

if(document.getElementsByClassName){

return document.getElementsByClassName(str);

}

*var* arrAllElements = document.getElementsByTagName("\*");

*var* arr = [];

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

// 需求：判断一个字符串中是否包含另外一个字符串

*var* arrClassName = arrAllElements[i].className.split(" "); // 把字符串变成数组

// 比较数组中的每一个元素，如果有一个和str相等，那么把所在的元素放进数组

for(*var* j=0;j<arrClassName.length;j++){

if(arrClassName[j] == str){

arr.push(arrAllElements[i]);

}

}

}

return arr;

},

/\*

调用者必须是获得者的兄弟节点

获取下一个兄弟节点

nextElementSibling、previousElementSibling; // 火狐谷歌IE9+支持获取下/上一个兄弟元素节点

nextSibling、previousSibing; // 主要是IE678支持, 但在火狐谷歌IE9+获取下/上一个节点（包括文本和换行）

\*/

nextSibling: *function*(*box*) {

return box.nextElementSibling || box.nextSibing;

},

/\*

获取上一个兄弟节点

\*/

preSibling: *function*(*box*) {

return box.previousElementSibling || box.previousSibing;

},

/\*

获得所有兄弟

\*/

siblings: *function*(*elm*) {

*var* arr = [];

*var* sibling = elm.parentNode.children;

for(*var* i = 0,len = sibling.length; i < len; i++) {

if(sibling[i] !== elm) arr.push(sibling[i]);

}

return arr;

},

/\*

调用者必须是获节点元素的父节点

firstElementChild、lastElementChild; // 火狐谷歌IE9+支持获取第一个/最后一个子元素节点

firstChild、lastChild; //主要是IE678支持, 但是在火狐谷歌IE9+获取第一个/最后一个子节点。（文本和换行）

\*/

firstChild: *function*(*box*) {

return box.firstElementChild || box.firstChild; // 获取第一个子节点

},

lastChild: *function*(*box*) {

return box.lastElementChild || box.lastChild; // 获取最后一个子节点

},

/\*

获取所有子节点

element.childNodes

W3C官方承认方法，是标准属性，它返回指定元素的子元素集合，包括HTML节点，所有属性，文本节点

火狐 谷歌等高本版会把换行也看做是子节点

element.children

非标准属性，它返回指定元素的子元素数组(最重要!!!)

但它只返回HTML节点，甚至不返回文本节点，虽然不是标准的DOM属性，

但它和innerHTML方法一样，得到了几乎所有浏览器的支持。

children在IE6/7/8中包含注释节点, 解决办法：把注释写在外边。

\*/

childrens: *function*(*box*) {

return box.childre;

},

/\*

父节点

调用者必须是获得节点的子节点

\*/

parentNode: *function*(*child*) {

return child.parentNode

},

/\*

清除选中的内容

window.getSelection() // IE9、Firefox、Safari、Chrome和Opera支持

document.selection // IE9以下支持

\*/

clearSelection: *function*() {

return window.getSelection ? window.getSelection().removeAllRanges() : document.selection.empty();

},

/\*

获得用户选择内容

\*/

getSelection: *function*() {

return window.getSelection ? window.getSelection().toString() : document.selection.createRange().text;

},

// 将生日转为周岁（PC必须用这个，否则IE8无法识别）

// 用于生日转化为周岁，精确到日

// utils.conversion('1991-09-01')

conversion: *function* (*strBirthday*) {

*var* returnAge;

*var* strBirthdayArr = strBirthday.split("-");

*var* birthYear = strBirthdayArr[0];

*var* birthMonth = strBirthdayArr[1];

*var* birthDay = strBirthdayArr[2];

*var* d = new Date();

*var* nowYear = d.getFullYear();

*var* nowMonth = d.getMonth() + 1;

*var* nowDay = d.getDate();

if (nowYear == birthYear) {

returnAge = 0; //同年 则为0岁

} else {

*var* ageDiff = nowYear - birthYear; //年之差

if (ageDiff > 0) {

if (nowMonth == birthMonth) {

*var* dayDiff = nowDay - birthDay; //日之差

if (dayDiff < 0) {

returnAge = ageDiff - 1;

} else {

returnAge = ageDiff;

}

} else {

*var* monthDiff = nowMonth - birthMonth; //月之差

if (monthDiff < 0) {

returnAge = ageDiff - 1;

} else {

returnAge = ageDiff;

}

}

} else {

returnAge = -1; //返回-1 表示出生日期输入错误 晚于今天

}

}

return returnAge; //返回周岁年龄

},

/\*

\* 用于生日转化为周岁，精确到日

\* utils.convertAge('1991-01-01')

\*\*/

convertAge:*function*(*params*) {

*let* now = new Date(),

date = new Date(params), // \* 1000

age,

o1 = {

Y: now.getFullYear(),

M: now.getMonth() + 1,

D: now.getDate()

},

o2 = {

Y: date.getFullYear(),

M: date.getMonth() + 1,

D: date.getDate()

};

age = o1.Y - o2.Y - ((o1.M < o2.M || o1.M === o2.M && o1.D < o2.D) ? 1 : 0);

return age;

},

convertDate: *function*(*params*) {

*let* fmt = 'yyyy-MM-dd h:m:s', // fmt = 'yyyy-MM-dd',

millisecond,

date,

o;

millisecond = parseInt(params, 10) \* 1000;

date = new Date(millisecond);

o = {

"M+": date.getMonth() + 1, //月份

"d+": date.getDate(), //日

"h+": date.getHours(), //小时

"m+": date.getMinutes(), //分

"s+": date.getSeconds(), //秒

"q+": Math.floor((date.getMonth() + 3) / 3), //季度

"S": date.getMilliseconds() //毫秒

};

if (/(y+)/.test(fmt))

fmt = fmt.replace(*RegExp*.$1, (date.getFullYear() + "").substr(4 - *RegExp*.$1.length));

for (*var* k in o)

if (new RegExp("(" + k + ")").test(fmt))

fmt = fmt.replace(*RegExp*.$1, (*RegExp*.$1.length == 1) ? (o[k]) : (("00" + o[k]).substr(("" + o[k]).length)));

return fmt;

//如果有填写打乱时间，就输出打乱时间

/\* if(vstartTime && vendTime) {

var randomRuslt = randomDate(vstartTime, vendTime, 'yyyy-MM-dd hh:mm:ss');

return randomRuslt;

}else {

return fmt;

} \*/

},

/\*

var vstartTime = $('input[name="startTime"]').val();

var vendTime = $('input[name="endTime"]').val();

\*/

randomDate: *function*(*s*, *e*, *f*) {

*var* num = 0;

*var* startTime = *Date*.parse(new Date(s));

*var* endTime = *Date*.parse(new Date(e));

*var* random, i, len = exportData ? exportData.length : 0;

*var* arr = [], result = [];

for (i = 0; i < len; i++) {

random = startTime + Math.random() \* (endTime - startTime);

result.push(DateFormatter(random, f || 'yyyy-MM-dd hh:mm:ss'));

}

result = result.sort();

result = result[num];

num += 1;

return result;

},

//根据身份证判断年龄和性别 22-50周岁

checkIdCard: *function* (*sId*) {

*var* iSum = 0

*var* info = ""

*var* sBirthday = sId.substr(6, 4) + "-" + *Number*(sId.substr(10, 2)) + "-" + *Number*(sId.substr(12, 2));

*var* sYear = sId.substr(6, 4)

*var* sSex = (sId.substr(16, 1) % 2 ? "男" : "女")

*var* d = new Date(sBirthday.replace(/-/g, "/"))

for (*var* i = 17; i >= 0; i--) iSum += (Math.pow(2, i) % 11) \* parseInt(sId.charAt(17 - i), 11)

return sBirthday + sSex

},

/\*

\* 前零位

\* utils.preZeroFill(2016801000000001, 8);

\* \*/

preZeroFill: *function*(*num*, *size*) {

*console*.log(Math.pow(10, size))

*console*.log(*Array*(size + 1).join('0'))

if (num >= Math.pow(10, size)) { //如果num本身位数不小于size位

return num.toString();

} else {

*var* \_str = *Array*(size + 1).join('0') + num;

return \_str.slice(\_str.length - size);

}

},

// 设置cookie有效期为指定时间

setCookie: *function*(*name*,*value*,*Deadline*,*callback*) {

// 获取当前日期对象

*var* curDate = new Date();

// 获取当前日期对应的时间戳

*var* curTime = curDate.getTime();

// 获取指定时间的时间戳

*var* endTime = $.convertTime(curDate,Deadline);

curDate.setTime(endTime);

// 计算出指定时间与当前时间的时间差

*var* disTime = endTime - curTime ;

// console.log(new Date(endTime));

// 设置cookie过期时间

document.cookie = name + '=' + value + '; expires=' + curDate.toGMTString();

// 在指定时间到达后执行回调

// setTimeout(callback,disTime);

},

// 获取指定时间的时间戳

convertTime: *function*(*nowDate*,*Deadline*){

// 分割参数Deadline

*var* \_dateArr = Deadline.split(':');

// 分别获取参数中对应的时、分、秒

*var* hours = parseInt(\_dateArr[0]);

*var* minutes = parseInt(\_dateArr[1]);

*var* seconds = parseInt(\_dateArr[2]);

// 设置对应时分秒

nowDate.setHours(hours);

nowDate.setMinutes(minutes);

nowDate.setSeconds(seconds);

// 获取当前天中指定时分秒对应的毫秒数

*var* result = *Date*.parse(nowDate); //返回 1970/1/1 午夜距离该日期时间的毫秒数

return result;

},

// 时间戳转化为日期

formatTime: *function* (*time*) {

*var* date = new Date(time \* 1000);

*var* year = date.getFullYear() + '/';

*var* month = date.getMonth() + 1 + '/';

*var* day = date.getDate();

*var* dateA = year + month + day;

return dateA;

},

// 提取身份证的日期

idcardBirthday: *function* (*idcard*) {

idcard = idcard.toString();

*var* idYear = idcard.substr(6,4) + '/';

*var* idMonth = idcard.substr(10,2) + '/';

*var* idDay = idcard.substr(12, 2);

*var* idAll = idYear + idMonth +idDay;

idAll = idAll.replace(/\/0/g, "/");

return idAll

},

// 用户生日以身份证为准

customerBirthday: *function* (*idCard*, *birthday*) {

*var* coverBirthday;

// 身份证有生日，就按显示身份证的值

//有值时

if (!(idCard === 'null ')) {

coverBirthday = utils.idcardBirthday(idCard);

}else {

coverBirthday = utils.formatTime(birthday);

}

return coverBirthday;

},

// 生日加零

addZero: *function* addZero(*num*) {

*var* arrBox = [];

*var* formatResult;

*var* numArr = num.split('/');

*console*.log(numArr[0].length);

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

*var* numStr;

numStr = numArr[i].length === 1 ? '0'+numArr[i]: numArr[i].toString();

arrBox.push(numStr);

}

formatResult = arrBox.join('-');

return formatResult;

},

}

// ===================

// 时间格式化

*Date*.prototype.format = *function*(*format*) {

*var* o = {

"M+": this.getMonth() + 1, //month

"d+": this.getDate(), //day

"h+": this.getHours(), //hour

"m+": this.getMinutes(), //minute

"s+": this.getSeconds(), //second

"q+": Math.floor((this.getMonth() + 3) / 3), //quarter

"S": this.getMilliseconds() //millisecond

};

if (/(y+)/i.test(format)) {

format = format.replace(*RegExp*.$1, (this.getFullYear() + "").substr(4 - *RegExp*.$1.length));

}

for (*var* k in o) {

if (new RegExp("(" + k + ")").test(format)) {

format = format.replace(*RegExp*.$1, *RegExp*.$1.length == 1 ? o[k] : ("00" + o[k]).substr(("" + o[k]).length));

}

}

return format;

};

new Date().format("yyyy-MM-dd hh:mm:ss"); // 2018-03-19 17:59:54

*module*.*exports* = {

name: 'shipeng',

modify: new Date().format("yyyy-MM-dd hh:mm:ss")

};

// ======================================

*var* browser={

versions:*function*(){

*var* u = navigator.userAgent; // 声明了浏览器用于 HTTP 请求的用户代理头的值, 只读

*var* app = navigator.appVersion; // 返回浏览器的平台和版本信息, 只读

return {//移动终端浏览器版本信息

trident: u.indexOf('Trident') > -1, //IE内核

presto: u.indexOf('Presto') > -1, //opera内核

webKit: u.indexOf('AppleWebKit') > -1, //苹果、谷歌内核

gecko: u.indexOf('Gecko') > -1 && u.indexOf('KHTML') == -1, //火狐内核

mobile: !!u.match(/AppleWebKit.\*Mobile.\*/), //是否为移动终端

ios: !!u.match(/\(i[^;]+;( U;)? CPU.+Mac OS X/), //ios终端

android: u.indexOf('Android') > -1 || u.indexOf('Linux') > -1, //android终端或者uc浏览器

iPhone: u.indexOf('iPhone') > -1 , //是否为iPhone或者QQHD浏览器

iPad: u.indexOf('iPad') > -1, //是否iPad

webApp: u.indexOf('Safari') == -1 //是否web应该程序，没有头部与底部

};

}(),

language:(navigator.browserLanguage || navigator.language).toLowerCase()

}

browser.versions.language;

if(!(browser.versions.mobile || browser.versions.ios || browser.versions.android || browser.versions.iPhone || browser.versions.iPad)){

window.location = "http://www.baidu.com/";

}

// -------------------------------分割线 -----------------------

if(device.mobile()){

$.ajax({

url:'/common/getUrl.html',

type:'POST',

data:{'url':top.location.href},

dataType:'json',

success:*function*(*data*){

if(data!=-1){ location.href = data; return; }

}

})

}

*var* \_ie6 = (navigator.appName == "Microsoft Internet Explorer" &&

navigator.appVersion .split(";")[1].replace(/[ ]/g,"")=="MSIE6.0");

*var* \_ie7 = (navigator.appName == "Microsoft Internet Explorer" &&

navigator.appVersion .split(";")[1].replace(/[ ]/g,"")=="MSIE7.0");

if(\_ie6 || \_ie7) {

location.href='http://www.baidu.com/';

}

// =======================================

// 验证

/\*

\* @Author: huoqishi

\* @Date: 2016-09-15 01:32:00

\* @Last Modified by: huoqishi

\* @Last Modified time: 2016-09-15 01:36:23

\*/

'use strict';

(*function*(*$*){

$.fn.extend({

formVerify:*function*(*o*){

o = o || {};

//正则

*var* reg = {

//手机号码

"phone":/^(13[0-9]|14[0-9]|15[0-9]|18[0-9])\d{8}$/i,

//邮箱

"email": /^([a-zA-Z0-9\_\.\-])+\@(([a-zA-Z0-9\-])+\.)+([a-zA-Z0-9]{2,4})+$/,

//密码，只能匹配大小写+数字6-18

"pass": /\w{5,17}/g,

//第二次输入密码验证

"rePass": /\w{5,17}/g,

//验证码，只能是6位数字

"authCode":/^\w{4,}$/,

//重置码

"resetcode":/^\w{4,}$/,

//邮箱或者手机号码

"phoneOrEmail": [/^(13[0-9]|14[0-9]|15[0-9]|18[0-9])\d{8}$/i,/^([a-zA-Z0-9\_\.\-])+\@(([a-zA-Z0-9\-])+\.)+([a-zA-Z0-9]{2,4})+$/]

},

//默认参数配置

defaults = {

email:{

noNullInfo:'您输入的邮箱格式有误，请重新输入！',

nullInfo:"邮箱不能为空！",

// successInfo:"邮箱验证成功！"

successInfo: ''

},

pass:{

noNullInfo:'密码长度必须在6~18之间，只能包含字符、数字和下划线！',

nullInfo:"密码不能为空！",

// successInfo:"密码验证成功！"

successInfo: ''

},

rePass:{

noNullInfo:'两次密码输入不一致！',

nullInfo:"密码不能为空！",

// successInfo:"密码验证成功！"

successInfo: ''

},

authCode:{

noNullInfo:'您输入的验证码有误，请重新输入！',

nullInfo:"验证码不能为空！",

// successInfo:"验证码验证成功！"

successInfo: ''

},

phone:{

noNullInfo:'您输入的手机号码有误，请重新输入！',

nullInfo:"手机号码不能为空！",

// successInfo:"手机号码验证成功！"

successInfo: ''

},

resetcode:{

noNullInfo:'您输入的重置码有误，请重新输入！',

nullInfo:"重置码不能为空！",

// successInfo:"重置码验证成功！"

successInfo: ''

},

phoneOrEmail:{

noNullInfo:'您输入的手机号码或邮箱有误，请重新输入！',

nullInfo:"手机号码或邮箱不能为空！",

// successInfo:"手机号码或邮箱验证成功！"

successInfo: ''

}

};

return this.each(*function*(*index*,*element*){

$.extend(defaults,o);

*var* successInfo = defaults[o.verify].successInfo,

nullInfo = defaults[o.verify].nullInfo,

noNullInfo = defaults[o.verify].noNullInfo,

verify = reg[defaults.verify];

//验证手机或者邮箱

if(o.verify == 'phoneOrEmail'){

//非法验证

if(verify[0].test($(element).val()) || verify[1].test($(element).val())){

$(element).nextAll('.tip').show().removeClass('error').addClass('success').html(successInfo);

//非空验证

}else if($(element).val() === ''){

$(element).nextAll('.tip').show().removeClass('success').addClass('error').html(nullInfo);

//成功验证

}else{

$(element).nextAll('.tip').show().removeClass('success').addClass('error').html(noNullInfo);

}

}else

//其它验证

//非法验证

if(verify.test($(element).val())){

$(element).next('.tip').show().removeClass('error').addClass('success').html(successInfo);

//非空验证

}else if($(element).val() === ''){

$(element).nextAll('.tip').show().removeClass('success').addClass('error').html(nullInfo);

//成功验证

}else{

$(element).nextAll('.tip').show().removeClass('success').addClass('error').html(noNullInfo);

}

//验证两次密码输入是否一致

if(o.verify === 'rePass'){

if($(element).val() === ''){

$(element).next('.tip').show().removeClass('success').addClass('error').html(nullInfo);

}else if($(this).val()!==$('#Pass').val()){

$(element).next('.tip').show().removeClass('success').addClass('error').html(noNullInfo);

}else if(verify.test($(element).val())){

$(element).next('.tip').show().removeClass('error').addClass('success').html(successInfo);

}

}

});

}

});

})(jQuery);

// =========================================

*var* answerArr1 = ["有买过", "没，但有社保", "社保和商保都有"];

*function* getRandomNum(*Min*,*Max*) {

*var* Range = Max - Min;

return Math.floor(Min + Math.random() \* Range);

}

*function* getAnswer(*answerArr*,*Min*,*Max*) {

*var* num = getRandomNum(Min,Max);

return answerArr[num];

}

//getAnswer(answerArr1, 0, 3)

// =======================================

*var* question01Obj = {

1: '1个',

2: '2个或以上',

3: '没有'

};

*function* setQuestion01(*method*) {

if (!isNaN(method)) {

return question01Obj[method];

}

};

// setQuestion01(this.question01);

// =======================================

if(window.XDomainRequest) {

*var* param = "name=" + nameVal +

"&gender=" + sexVal +

"&mobile=" + mobileVal +

"&email=" + emailVal +

"&birthday=" + birthdayVal +

"&userProvinceId=" + userProvinceIdVal +

"&userCityId=" + userCityIdVal;

*var* xdr = new XDomainRequest();

//xdr.setRequestHeader('Content-Type','application/x-www-form-urlencoded');

// 根据指定的方法（GET或POST）和URL，打开请求。

//xdr.open("POST", 'http://api\_hdyl.vrm.cn/code/'+ corpId +'/' + activityId);

xdr.open("POST", 'http://192.168.1.183:8082/rms-back/code/1/1');

// 当请求中发送方法和onload事件中有进展时的处理程序。

xdr.onprogress = *function* () {};

// 当请求超时时的事件处理程序。

xdr.ontimeout = *function* () { };

// 当请求发生错误时的处理程序。

xdr.onerror = *function* (*res*) {

*console*.log(res);

};

// 当服务器端的响应被完整接收时的处理程序。

xdr.onload = *function*(*res*) {

*var* text = xdr.responseText; //获取仅仅是字符串

*var* json\_obj = JSON.parse(text); //把字符串转化成JSON对象

*console*.log(json\_obj);

if(json\_obj.retCode == "10000") {

*console*.log(json\_obj.retCode);

// 初始化弹层引流

initPopYLLayerData(json\_obj.data.type0);

// 初始化尾部引流

initTailYLLayerData(json\_obj.data.type1);

// 初始化抽奖页面

initLuckLayerData(json\_obj.data.type2);

} else {

*console*.log(res);

}

}

// 发送请求。POST的数据会在该方法中被指定。

setTimeout(*function* () {

xdr.send("birthday=1990-6-17&mobile=13431093388&gender=M");

}, 0);

}

export default utils