**小程序验证码的实现**

1. **实现原理  
   向后端发送请求，后端处理发送验证码，获取验证码，通过画图函数画出含有数字，线条，扰乱的图像。当再一次点击图片时，重新发送一次后端请求。**
2. **实例教学  
   2.1 canvas画图，在WXML页面创建：**<view class="code" bindtap="reflesh">

<canvas canvas-id="my-canvas" style="width: 260rpx; height: 86rpx;"/>

</view>

Canvas-id 属性标志，便于JS获取该元素，由于小程序没有文档和窗口对象，因此不能进行DOM,BOM操作。

2.2页面js  
let app = getApp()

onLoad: function (options) {

var str = app.globalData.captchaTxt

//获取app.js中获取的验证码

app.drawPic(str)

//调用app中画图方法

},

2.3app.js

/\*验证码\*/

guid() {

function S4() {

return (((1 + Math.random()) \* 0x10000) | 0).toString(16).substring(1);

}

return (S4() + S4() + "-" + S4() + "-" + S4() + "-" + S4() + "-" + S4() + S4() + S4());

},

// 获取验证码

genCaptcha() {

let captchaId = wx.getStorageSync('captchaId')

if (!captchaId) {captchaId = this.guid()}

wx.setStorageSync('captchaId', captchaId)

console.log('cid',captchaId)

wx.request({

url: `${host2}/m/captcha`,

data: {captchaId},

method: 'POST', // OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT

header: { 'content-type': 'application/x-www-form-urlencoded', 'Authorization': 'Bearer ' + wx.getStorageSync('jwt') }, // 设置请求的 header

success: res => {

// success

console.log('set验证码 ', res)

if (res.data.result\_code == 1) {

this.globalData.captchaId = captchaId

this.globalData.captchaTxt = res.data.result

var str = this.globalData.captchaTxt

this.drawPic(str)

console.log('最新str', str)

}

}

})

},

// 生成一个随机数

randomNum(min, max) {

return Math.floor(Math.random() \* (max - min) + min);

},

// 随机颜色

randomColor(min, max) {

var r = this.randomNum(min, max);

var g = this.randomNum(min, max);

var b = this.randomNum(min, max);

return "rgb(" + r + "," + g + "," + b + ")";

},

/\*\*绘制验证码图片\*\*/

drawPic(str) {

var width = 120

var height = 50

var ctx = wx.createCanvasContext('my-canvas')

// 绘制背景色

let color = this.randomColor(180, 240)

ctx.setFillStyle(color)

ctx.rect(0, 0, width, height)

ctx.fill()

console.log('str', str)

// 绘制英文

for (var i = 0; i < 4; i++) {

var txt = str[i];

var fillStyle = this.randomColor(50, 160) //随机生成字体颜色

var font = this.randomNum(30, 40) //随机生成字体大小

var x = 10 + i \* 26;

var y = this.randomNum(35, 40)

var deg = this.randomNum(-25, 25)

ctx.setFillStyle(fillStyle)

ctx.setFontSize(font)

//修改坐标原点和旋转角度

ctx.translate(x, y);

ctx.rotate(deg \* Math.PI / 180);

ctx.fillText(txt, 0, 0);

//恢复坐标原点和旋转角度

ctx.rotate(-deg \* Math.PI / 180);

ctx.translate(-x, -y);

}

// 绘制干扰线

for (var i = 0; i < 3; i++) {

var strokeStyle = this.randomColor(40, 180)

ctx.setFillStyle(strokeStyle)

ctx.beginPath();

let x = this.randomNum(0, width)

let y = this.randomNum(0, height)

ctx.moveTo(x, y);

ctx.lineTo(this.randomNum(x - 30, x + 30), this.randomNum(y - 30, y + 30));

ctx.stroke();

}

// 绘制干扰点

for (var i = 0; i < 10; i++) {

var Style = this.randomColor(0, 255);

ctx.setFillStyle(Style)

ctx.beginPath();

ctx.arc(this.randomNum(0, width), this.randomNum(0, height), 1, 0, 2 \* Math.PI);

ctx.fill();

}

ctx.draw()

},