如果你还在使用canvas绘制海报生成，那么你OUT了！
那么请你耐心的看下以下，相信以后的海报需求，你会很容易说So easy!
废话不多说，直接上主题

1.先在小程序中创建文件夹canvasdrawer
然后就扒扒扒代码
wxml:

```
<canvas canvas-id="canvasdrawer" style="width:{{width}}px;height:{{height}}px;" class="board"
wx:if="{{showCanvas}}"></canvas>
```

wxss:

```
.board {
  position: fixed;
  top: 2000rpx;
}
```

json:

```
{
  "component": true
}
```

js:

```
/* global Component wx */

Component({
  properties: {
    painting: {
      type: Object,
      value: {view: []},
      observer (newVal, oldVal) {
        if (!this.data.isPainting) {
          if (JSON.stringify(newVal) !== JSON.stringify(oldVal)) {
            if (newVal && newVal.width && newVal.height) {
              this.setData({
                showCanvas: true,
                isPainting: true
              })
              this.readyPigment()
            }
        } else {
          if (newVal && newVal.mode !== 'same') {
            this.triggerEvent('getImage', {errMsg: 'canvasdrawer:samme params'})
          }
        }
      }
    }
  }
},
data: {
  showCanvas: false,

  width: 100,
  height: 100,

  tempFileList: [],

  isPainting: false
},
ctx: null,
cache: {},
ready () {
  wx.removeStorageSync('canvasdrawer_pic_cache')
  this.cache = wx.getStorageSync('canvasdrawer_pic_cache') || {}
  this.ctx = wx.createCanvasContext('canvasdrawer', this)
},
methods: {
  readyPigment () {
    const { width, height, views } = this.data.painting
    this.setData({
      width,
      height
    })
    const inter = setInterval(() => {
      if (this.ctx) {
        clearInterval(inter)
        this.ctx.clearActions()
        this.ctx.save()
        this.getImagesInfo(views)
      }
    }, 100)
  },
getImagesInfo (views) {
  const imageList = []
  for (let i = 0; i < views.length; i++) {
    if (views[i].type === 'image') {
      imageList.push(this.getImageInfo(views[i].url))
    }
  }

  const loadTask = []
  for (let i = 0; i < Math.ceil(imageList.length / 8); i++) {
    loadTask.push(new Promise((resolve, reject) => {
      Promise.all(imageList.splice(i * 8, 8)).then(res => {
        resolve(res)
      }).catch(res => {
        reject(res)
      })
    }))
  }
  Promise.all(loadTask).then(res => {
    let tempFileList = []
    for (let i = 0; i < res.length; i++) {
      tempFileList = tempFileList.concat(res[i])
    }
    this.setData({
      tempFileList
    })
    this.startPainting()
  })
},
startPainting () {
  const { tempFileList, painting: { views } } = this.data
  console.log(tempFileList)
  for (let i = 0, imageIndex = 0; i < views.length; i++) {
    if (views[i].type === 'image') {
      this.drawImage({
        ...views[i],
        url: tempFileList[imageIndex]
      })
      imageIndex++
    } else if (views[i].type === 'text') {
      if (!this.ctx.measureText) {
        wx.showModal({
          title: '提示',
          content: '当前微信版本过低，无法使用 measureText 功能，请升级到最新微信版本后重
试。'
        })
        this.triggerEvent('getImage', {errMsg: 'canvasdrawer:version too low'})
        return
      } else {
        this.drawText(views[i])
      }
    } else if (views[i].type === 'rect') {
      this.drawRect(views[i])
    }
  }
  this.ctx.draw(false, () => {
    wx.setStorageSync('canvasdrawer_pic_cache', this.cache)
    const system = wx.getSystemInfoSync().system
    if (/ios/i.test(system)) {
      this.saveImageToLocal()
    } else {
      // 延迟保存图片，解决安卓生成图片错位bug。
      setTimeout(() => {
        this.saveImageToLocal()
      }, 800)
    }
  })
},
// drawImage (params) {
//   this.ctx.save()
//   const { url, top = 0, left = 0, width = 0, height = 0, borderRadius = 0, deg = 0 } = params
//   // if (borderRadius) {
//   //   this.ctx.beginPath()
//   //   this.ctx.arc(left + borderRadius, top + borderRadius, borderRadius, 0, 2 * Math.PI)
//   //   this.ctx.clip()
//   //   this.ctx.drawImage(url, left, top, width, height)
//   // } else {
//   if (deg !== 0) {
//     this.ctx.translate(left + width/2, top + height/2)
//     this.ctx.rotate(deg * Math.PI / 180)
//     this.ctx.drawImage(url, -width/2, -height/2, width, height)
//   } else {
//     this.ctx.drawImage(url, left, top, width, height)
//   }
```

```
  //   // }
  //   this.ctx.restore()
  // },
  drawImage(params) { this.ctx.save(); const { url, top = 0, left = 0, width = 0, height = 0,
borderRadius = 0 } = params; if (borderRadius) { let d = borderRadius * 2; let cx = left +
borderRadius; let cy = top + borderRadius; this.ctx.beginPath(); this.ctx.arc(cx, cy, borderRadius,
0, 2 * Math.PI); this.ctx.fill(); this.ctx.clip(); this.ctx.drawImage(url, left, top, d, d); } else {
this.ctx.drawImage(url, left, top, width, height); } this.ctx.restore(); },
  drawText (params) {
    this.ctx.save()
    const {
      MaxLineNumber = 2,
      breakWord = false,
      color = 'black',
      content = '',
      fontSize = 16,
      top = 0,
      left = 0,
      lineHeight = 20,
      textAlign = 'left',
      width,
      bolder = false,
      textDecoration = 'none'
    } = params

    this.ctx.beginPath()
    this.ctx.setTextBaseline('top')
    this.ctx.setTextAlign(textAlign)
    this.ctx.setFillStyle(color)
    this.ctx.setFontSize(fontSize)

    if (!breakWord) {
      this.ctx.fillText(content, left, top)
      this.drawTextLine(left, top, textDecoration, color, fontSize, content)
    } else {
      let fillText = ''
      let fillTop = top
      let lineNum = 1
      for (let i = 0; i < content.length; i++) {
        fillText += [content[i]]
        if (this.ctx.measureText(fillText).width > width) {
          if (lineNum === MaxLineNumber) {
```

```
          if (i !== content.length) {
            fillText = fillText.substring(0, fillText.length - 1) + '...'
            this.ctx.fillText(fillText, left, fillTop)
            this.drawTextLine(left, fillTop, textDecoration, color, fontSize, fillText)
            fillText = ''
            break
          }
        }
        this.ctx.fillText(fillText, left, fillTop)
        this.drawTextLine(left, fillTop, textDecoration, color, fontSize, fillText)
        fillText = ''
        fillTop += lineHeight
        lineNum ++
      }
    }
    this.ctx.fillText(fillText, left, fillTop)
    this.drawTextLine(left, fillTop, textDecoration, color, fontSize, fillText)
  }

  this.ctx.restore()

  if (bolder) {
    this.drawText({
      ...params,
      left: left + 0.3,
      top: top + 0.3,
      bolder: false,
      textDecoration: 'none'
    })
  }
},
drawTextLine (left, top, textDecoration, color, fontSize, content) {
  if (textDecoration === 'underline') {
    this.drawRect({
      background: color,
      top: top + fontSize * 1.2,
      left: left - 1,
      width: this.ctx.measureText(content).width + 3,
      height: 1
    })
  } else if (textDecoration === 'line-through') {
    this.drawRect({
      background: color,
      top: top + fontSize * 0.6,
      left: left - 1,
      width: this.ctx.measureText(content).width + 3,
      height: 1
    })
  }
},
drawRect (params) {
  this.ctx.save()
  const { background, top = 0, left = 0, width = 0, height = 0 } = params
  this.ctx.setFillStyle(background)
  this.ctx.fillRect(left, top, width, height)
  this.ctx.restore()
},
getImageInfo (url) {
  return new Promise((resolve, reject) => {
    if (this.cache[url]) {
      resolve(this.cache[url])
    } else {
      const objExp = new RegExp(/^http(s)?:\/\/([\w-]+\.)+[\w-]+(\/[\w- .\/?%&=]*)?/)
      if (objExp.test(url)) {
        wx.getImageInfo({
          src: url,
          complete: res => {
            if (res.errMsg === 'getImageInfo:ok') {
              this.cache[url] = res.path
              resolve(res.path)
            } else {
              this.triggerEvent('getImage', {errMsg: 'canvasdrawer:download fail'})
              reject(new Error('getImageInfo fail'))
            }
          }
        })
      } else {
        this.cache[url] = url
        resolve(url)
      }
    }
  })
},
saveImageToLocal () {
      const { width, height } = this.data
      wx.canvasToTempFilePath({
        x: 0,
        y: 0,
        width,
        height,
        canvasId: 'canvasdrawer',
        complete: res => {
          if (res.errMsg === 'canvasToTempFilePath:ok') {
            this.setData({
              showCanvas: false,
              isPainting: false,
              tempFileList: []
            })
            this.triggerEvent('getImage', {tempFilePath: res.tempFilePath, errMsg: 'canvasdrawer:ok'})
          } else {
            this.triggerEvent('getImage', {errMsg: 'canvasdrawer:fail'})
          }
        }
      }, this)
    }
  }
})
```

2.拔完之后直接新建test页面,搞一搞，玩一玩。
页面json文件直接引入"canvasdrawer": "/components/canvasdrawer/canvasdrawer"
wxml:

```
<canvasdrawer painting="{{painting}}"  bind:getImage="eventGetImage"/>
```

js:

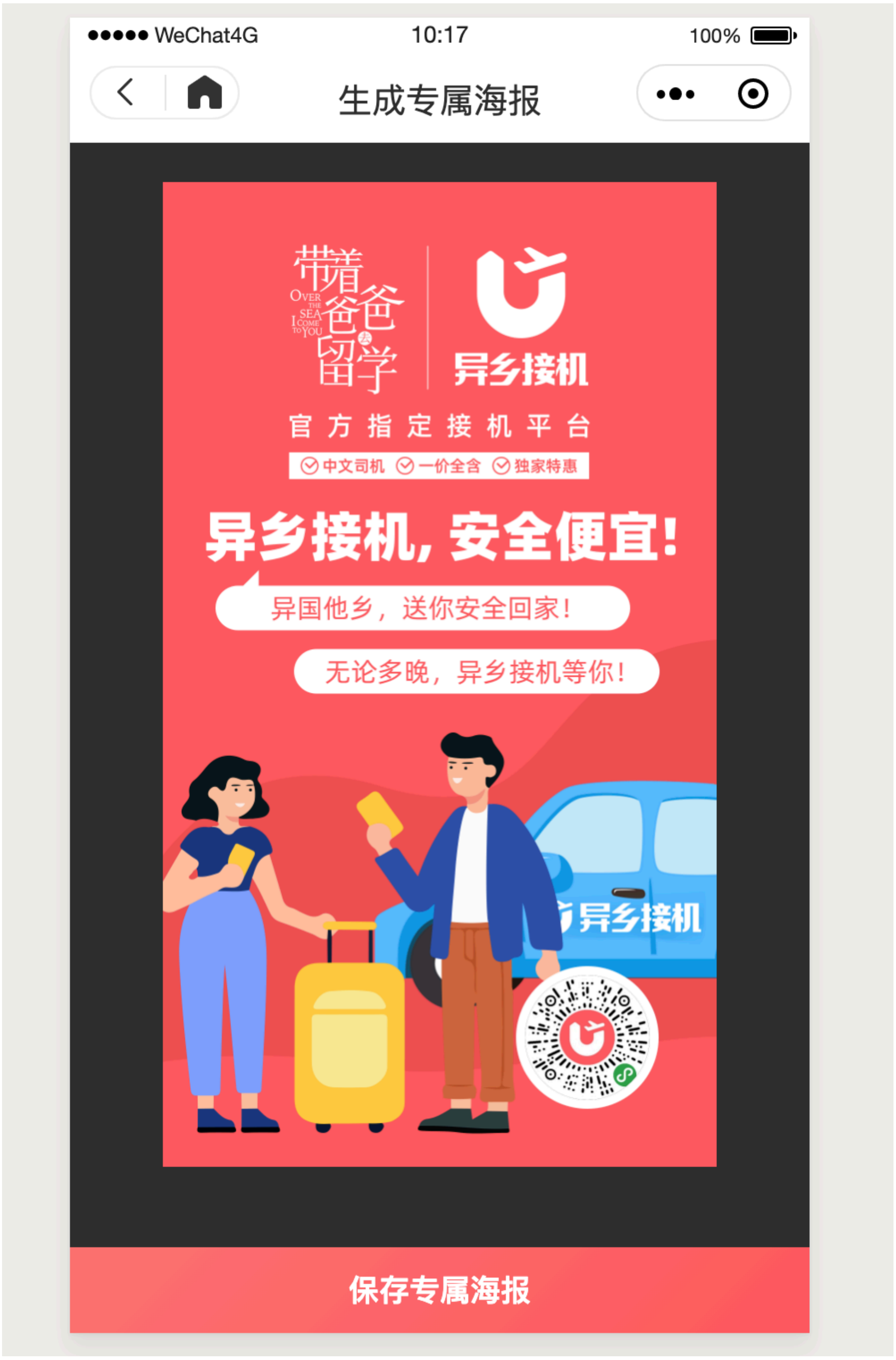
```
data:{
 painting: {
              width: 1242,
              height: 2208,
              clear: true,
              views: [{
              type: 'image',

url:"https://pic.uhouzz.com/AssistanceActivityImages/4d/b34f218b7271bc8ca67fa94c7053f17fc066da.jpg",
top: 0,
                left: 0,
              width: 1242,
              height: 2208,
               },
               {
                type: 'image',
                url: "https://img.uhomes.com/qrcode/wxapp/df/fe32c44471cb029d29ced53f33bfd20b7fa6ff.jpg",
top: 1778,
                left: 812,
                width: 280,
                height: 280
                borderRadius: 140
               },



              ]
             };
}
//TODO:组件生成出来的为BASE64
 eventGetImage:function(event) {
  console.log(event)
  const {
   tempFilePath,
   errMsg
  } = event.detail
  if (errMsg === 'canvasdrawer:ok') {
   this.setData({
    shareImage: tempFilePath
   })
  }
 },
```

3.嚯嚯哈嚯，至此打完收工！让我看一下效果吧



4.从此以后妈妈再也不担心我生成海报了！

5.相关API

数据对象的第一层需要三个参数: `width`、`height`、`mode`、`views`。配置中所有的数字都是没有单位的。这就意味着 `canvas` 绘制的是一个比例图。具体显示的大小直接把返回的图片路径放置到 `image` 标签中即可。
`mode` 可选值有 `same`, 默认值为空，常规下尽量不要使用。如要使用请看 Q&A的第1点。
当前可以绘制3种类型的配置: `image`、`text`、`rect`。配置的属性基本上使用的都是 `css` 的驼峰名称，还是比较好理解的。

image（图片）

属性	含义	默认值
url	绘制的图片地址，可以是本地图片，如：`/images/1.jpeg`	
top	左上角距离画板顶部的距离	
left	左上角距离画板左侧的距离	
width	要画多宽	0
height	要画多高	0

text（文本）

属性	含义	默认值
content	绘制文本	''（空字符串）
color	颜色	black
fontSize	字体大小	16
textAlign	文字对齐方式	left
lineHeight	行高，只有在多行文本中才有用	20
top	文本左上角距离画板顶部的距离	0
left	文本左上角距离画板左侧的距离	0
breakWord	是否需要换行	false
MaxLineNumber	最大行数，只有设置 `breakWord: true` ，当前属性才有效，超出行数内容的显示为...	2
width	和 `MaxLineNumber` 属性配套使用， `width` 就是达到换行的宽度	
bolder	是否加粗	false
textDecoration	显示中划线、下划线效果	none

rect (矩形，线条)

属性	含义	默认值
background	背景颜色	black
top	左上角距离画板顶部的距离	
left	左上角距离画板左侧的距离	
width	要画多宽	0
height	要画多高	0

6.本文技术非原创,写此目的仅为全面提高前端人员加快开发模式，提高工作效率

originalAuthor:https://github.com/kuckboy1994/mp_canvas_drawer

created by whq361