「微信小程序」入坑总结：相关 demo：仿 one

前段时间去了广州参加了关于「微信小程序」的技术沙龙，整场下来收获颇多。实际上微信小程序刚刚发布出来的时候就关注，弄了破解了的开发工具，跑了个官方的 Demo。

听完回来之后就想着挖个相关的坑，正好看到手机上的 「ONE · 一个」。上网找到了 ONE 的接口，打算搞一个仿照 ONE 官方 APP 的小程序，最后就有了 weapp-one 这个项目。

关于小程序

微信小程序不是 Web 也不是 Native，尝试之后给我的感觉有点像是 weex。因为之前尝试 weex 的时候是下载一个 APP，然后通过应用内扫码来运行对应的应用。这一点上小程序类似。

在沙龙上有位分享嘉宾有提到，小程序实际上是在 Webview 之上再加上一层原生的组件。例如底部的 tabbar 以及顶部的 navbar 就是原生的。还有小程序的 map 组件，是调用原生的 map 组件。

请求限制

一开始就遇到了问题，ONE 官方的接口是 http 的，而且是 8000 这个端口。微信小程序对发送的请求有限制，必须是 https（在本地开发可以用 http），而且不能带端口号。

所以我就使用了比较暴力的方法，把开发工具中对应用来限制的代码给注释掉了，反正只是用来体验一下小程序开发，所以就无所谓了。

添加页面

在小程序中添加页面都需要将路径添加到 `app.json` 中的 `page` 项中，否则会找不到页面。与 Vue 类似，每个小程序页面包含 `page.js`，`page.wxml`，`page.wxss`，分别对应脚本，模板以及样式。

还可以添加一个 `page.json` 对页面进行单独配置，比如配置 `navigationBarTitleText` 来改变导航上的文字显示，诸如此类的对页面的窗口表现配置。

奇怪的组件

view、text

在 wxml 里面写的是类似 HTML 的标签，标准的 HTML 标签是无法使用的。只能使用微信小程序官方的组件，组件对应有一些属性或事件可以调用。有类似于 `<div>` 的 `<view>` 组件，基本上页面上的组件都会被 `<view>` 包裹住。

然后文字方面是使用 `<text>`，虽然直接显示文字也没什么问题。不过我还是把所有的文字都加上了 `<text>`标签。

image

说起组件，微信小程序里最让我不爽的就是 `<image>` 这个组件了。给这个组件一个图片地址之后，默认的样式不是图片的大小，而是固定的 300px * 225px。与 HTML 中的 `<img>` 完全不同，用起来有点不舒服。

`<image>` 组件还提供了不同显示方式的 mode，不过用起来还是觉得怪怪的。

audio

音频播放的话有 `<audio>` 这个组件，但是这个组件的样式好像是固定的，类似于在网页也加上网易云音乐的外链那样。不过幸好有音频播放相关的 api 可以用，这样就可以当用户触发某些操作的时候播放音频，即可实现播放按钮点击后播放音频。

微信小程序在播放音频的时候，开发工具上会出现对应的音乐栏，可以对播放的音频进行播放/暂停。猜猜在真机上使用时，会在通知栏出现音频控制。上面会显示音频的作者以及歌曲名称。但是在实际使用中有一点比较尴尬，使用调用 api 的方式播放音频没有设置音频作者的选项。详情可以见 音乐播放控制 · 小程序

video

相较与 HTML5 中的 `<video>` 标签，微信小程序中的 `<video>` 组件缺失了很多东西，例如 `loop` 与 `poster`。也就是说无法在用户点击播放按钮前显示特定的图像，只能是显示视频最开头。

`<video>` 组件同样有默认的尺寸 300px * 225px，但没有 `<image>` 组件中的显示模式的设置。未全屏状态下，在不同的分辨率上，上下或者左右可能会存在黑边。

特殊的 rpx

微信小程序中有特有的一个尺寸单位 —— rpx，1rpx 表示屏幕宽度的 1/750 大小，也就是说，100% 宽度就是 750rpx。还有一个会被忽略的尺寸单位 —— rem，与 Web 中的 rem 不同，1rem 表示屏幕宽度的 1/20 大小。

在高度上使用 rpx 的话，也会根据屏幕的宽度的大小而改变。我更倾向与宽度使用 rpx 而高度使用 rem，不过其实比较喜欢用 px 来写样式，这可能算是一个不好的习惯。

实现滑至最右切换页面

在仿照 ONE 写微信小程序的时候遇到了一个问题，使用滑块视图组件 `<swiper>` 去显示每日图文，一共 10 个图文，当滑动至最右时切换到选择往期列表的页面。如下图所示：

但是 `<swiper>` 组件并没有提供对应的事件，使用就自己实现了一个。主要是使用滑块视图每一次切换视图时都会触发一个 `bindchange` 事件，还有设置滑块视图显示位置的 `current` 属性。

在 `<swiper>` 组件中添加一个空的 `<swiper-item>` 子组件，当滑动到这个空的子组件的时候使用 `wx.navigateTo` 这个 api 去切换到往期列表这个页面。同时将滑块视图的显示位置设置到倒数第二个，即最后一个非空的子组件。

这样就实现了滑动至最右切换页面的功能，同时在点击返回的时候显示的也是滑块视图中的最后一个有内容的子组件。具体实现代码如下：

```
Page({
  data: {
    current: 0
  },
  // ......
  // more code
  handleChange: function (e) {
    let current = e.detail.current
    let length = this.data.vols.length
    if (current === length) {
      this.setData({
```

```
            current: length
        })
        wx.navigateTo({
            url: '../history/history?page=index',
            success: () => {
                this.setData({
                    current: length - 1
                })
            }
        })
    }
}
})

```

这里使用了两次 `this.setData` 是因为在第二次执行上面的滑动切换页面再返回的时候，显示的子组件并不是最后一个非空的子组件，而是最后的那个空子组件。大概原因是因为第二次执行这个操作的时候，`current` 并

没有更新。

所以解决的方案是在每次修改 `current` 之前修改一次它的值，使得后面修改 `current` 值时会触发视图的更新。