微信小程序开发入门练习篇

前言

直接开始，简单理解就是微信作为原生层，我们的应用作为网页，是一种 hybird 的开发方式，唯一不同的是，在这个平台上必须遵守微信的一些设计规范、运营规范等东西。

官方资料

https://mp.weixin.qq.com/debug/wxadoc/dev/?t=1474644088899

开发工具

官方已出正版，可直接使用无 appid 进行体验

下载地址

使用的开发元素

1. JavaScript （交互、数据等脚本）
2. wxml （页面结构、组件）
3. wxss（页面样式）

`wxml` 和 `wxss` 是新的文件格式，不用理解，就映射成 `html` 和 `css` 即可，但是不同的是，有一套自己的标签和支持的范围（比如 css 部分支持度有限），详情可以看上面的官方资料。

项目的结构

下载到 DEMO 过后，就会有一个基本的项目结构，相当简单。

```
├── app.js
├── app.json
├── app.wxss
├── pages
│   ├── index
│   └── logs
└── utils
    └── util.js
```

其他就不看了，直接看 `app.json` 这一个文件即可，这是一个全局 app 的配置文件，具体详情：文档

有很多配置，具体看文档，这里主要说一下三个配置：

- 导航、标题部分 使用的是原生 header, 只能改变: 导航栏的颜色，不能修改其中的内容。导航，需要开发者自己控制（微信规定，页面路径只能是五层）。
- pages 部分 默认加载第一个页面，其余靠跳转
- 底部导航（tabBar） 只能配置最少 2 个、最多 5 个 tab。

其余配置就不需要怎么了解了，根据后面的小栗子来

hello world

- app.json

```
{
  "pages":[
    "pages/index/index"
  ],
  "window":{
    "backgroundTextStyle":"light",
    "navigationBarBackgroundColor": "#fff",
    "navigationBarTitleText": "WeChat",
    "navigationBarTextStyle":"black"
  }
}
```

- app.js

```
App({}) // 暂时先什么都不管
```

- pages/index/index.js

```
Page({
  data: {
    hello: "Hello World",
  }
});
```

- pages/index/index.wxml

```
<view class="container">
    <text>{{hello}}</text>
</view>
```

输出就不用截图了，就是替换 {{hello}} => 'Hello World'

可以看出，`app.js` 是启动的入口文件，然后 `pages` 目录下面都有一个 Page 对象来做页面封装，然后再渲染到页面中，启动结构: App => Pages 的模式，典型的类 App 开发（ manager => activity）。

使用过 angularjs 或者 vue 等框架的直接就可以上手了，MVVM 模式，但是唯一不一样的是提供手动触发界面渲染，使用 `this.setData()` 方法，具体查看 API 或者跟我一起看后面的栗子。

一个简单的静态表单

在微信提供的这套框架中，提供了很多组件，这里用一个静态表单的代码来感性认识一下，具体的还得你看 API 了哦。

> 注：栗子都是修改 `app.json` 中的 `pages` 配置，直接默认预览，因为不想受 navigate 的跳转影响，那个很简单，看 API 传参数即可。

要做的效果是：

- pages/form.wxml

```
<view class="page">
  <form class="form" catchsubmit="formSubmit">
    <view class="section">
      <view class="label">姓名</view>
      <input name="username" placeholder="请输入姓名" />
    </view>
    <view class="section">
      <view class="label">性别</view>
      <radio-group name="sex" >
        <label wx:for-items="{{sex}}"><radio value="{{item.key}}"
/>{{item.value}}</label>
      </radio-group>
    </view>
    <view class="section">
```

```
      <view class="label">爱好</view>
      <checkbox-group name="love">
        <label wx:for-items="{{love}}"><checkbox value="{{item.key}}"
/>{{item.value}}</label>
      </checkbox-group>
    </view>

    <view class="btn-area">
        <button type="primary" formType="submit">Submit</button>
    </view>
  </form>
</view>
```

- pages/form.js

```
var pageObject = {
  data: {
    sex: [],
    love: []
  },
  // 注意事件部分，提供常见的事件
  formSubmit: function(e) {
    console.log("表单数据为：", e.detail.value)
  }
};

// 生成数据
var radios = [{
  key: "1",
  value: "男"
}, {
  key: "2",
  value: "女"
}];
```

```
var checkboxs = [{
  key: "1",
  value: "篮球"
}, {
  key: "2",
  value: "乒乓球"
}, {
  key: "3",
  value: "足球"
}];

var len = radios.length;
for(var i = 0; i < len; ++i) {
  (function (_i) {
    pageObject.data["sex"].push(radios[_i]);
  })(i);
}

len = checkboxs.length;
for(var i = 0; i < len; ++i) {
  (function (_i) {
    pageObject.data["love"].push(checkboxs[_i]);
  })(i);
}

Page(pageObject);
```

静态的很简单，这里主要就是构建页面使用它的组件和风格，还是特别快的

CSS 部分

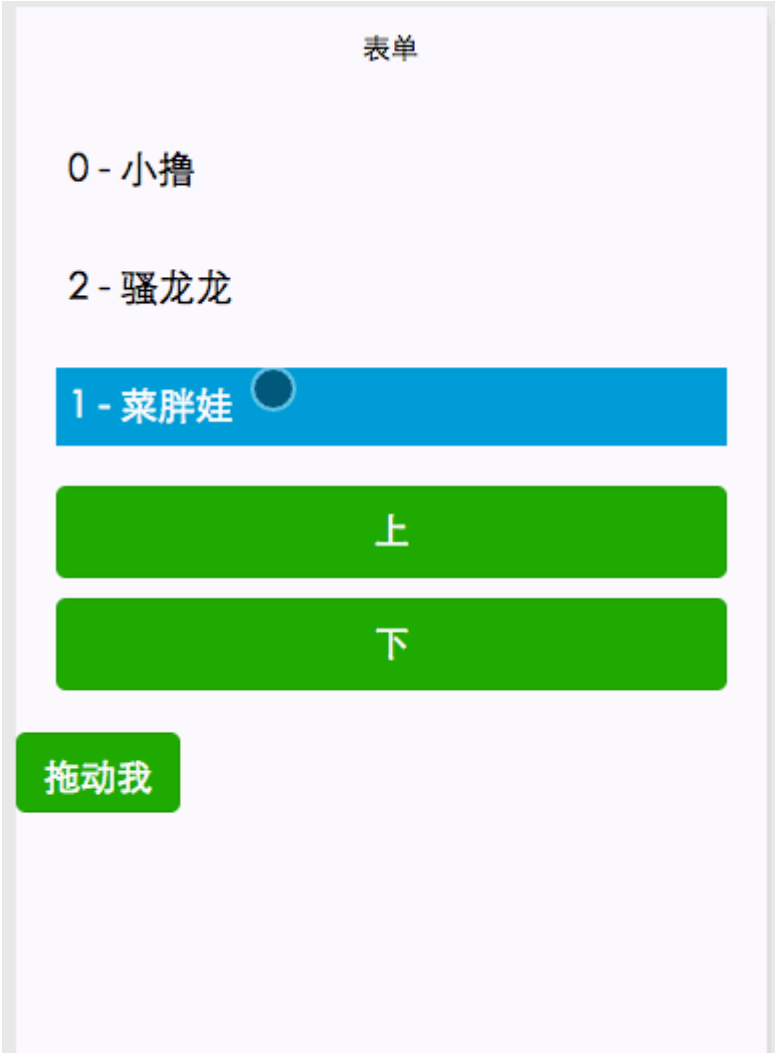
```
.form {
  display: block;
  padding: 0.6em;
  margin-top: -15px;
```

```
}

.section {
  margin-top: 15px;
}
.section .label {
  font-size: 14px;
  color: #333;
  line-height: 28px;
}

.btn-area {
  margin-top: 15px;
}

input {
  font-size: 14px;
  height: 28px;
  padding-left: 6px;
  border: 1px solid #D9D9D9;
  background: #fff;
}

input:focus {
  outline: 0 none;
}

label {
  margin-left: 10px;
  font-size: 14px;
  vertical-align: middle;
}
```

```
radio-group, checkbox-group {
  margin-left: -10px;
}

radio, checkbox {
  vertical-align: middle;
  margin-right: 6px;
}
```

稍微复杂交互的例子

有拖动（拖动效果先不管）直接样式更新以及简单的数据驱动的交互。



- pages/simple.wxml

```
<view class="main">
    <button class="btn"
      style="top:{{btnTop}};left:{{btnLeft}}"
      type="primary"
      data-name="我是 button."
      bindtap="tapClick"
```

```
      bindtouchstart="touchStart"
      bindtouchmove="touchMove">拖动我</button>

    <text data-id="{{item.id}}" class="text {{item.active ? "on" : ""}}"
bindtap="tapText" wx:for-items="{{values}}">{{item.id + " - " +
item.value}}</text>

    <view class="move">
      <button type="primary" bindtap="onPrev">上</button>
      <button type="primary" bindtap="onNext">下</button>
    </view>
</view>
```

- pages/simple.js

```
var pageObject = {
  data: {
    btnTop: 0,
    btnLeft: 0,
    move: 0,
    values: [{
      id: "0",
      value: "小撸",
      active: false
    }, {
      id: "1",
      value: "菜胖娃",
      active: false
    }, {
      id: "2",
      value: "骚龙龙",
      active: false
    }]
  },
  tapClick: function (e) {
```

```
},
touchStart: function (e) {
},
touchMove: function (e) {
  var touches = e.touches;
  var touch = touches[0];
  var pageY = touch.pageY + "px";
  this.setData({
    btnTop: pageY
  });
},
onPrev: function () {
  var move = this.data.move;
  if(move <= 0) {
    return false;
  }

  var values = this.data.values;

  var curValue = values[move];
  var prevValue = values[move - 1];

  values.splice(move - 1, 2, curValue, prevValue);

  this.setData({
    move: move - 1,
    values: values
  });
},
onNext: function () {
  var move = this.data.move;
  var values = this.data.values;
  if(move >= values.length - 1) {
```

```
    return false;
  }

  var curValue = values[move];
  var nextValue = values[move + 1];

  values.splice(move, 2, nextValue, curValue);

  this.setData({
    move: move + 1,
    values: values
  });
},
tapText: function (e) {
  var dataset = e.target.dataset || {};
  var id = e.target.dataset.id;

  if(id) {
    var index = -1;
    var newValues = this.data.values.map(function (d, i) {
      if(d.id === id) {
        d.active = !d.active;
        index = i;
      } else {
        d.active = false;
      }
      return d;
    });

    this.setData({
      move: index,
      values: newValues
    });
```

```
    }
  }
};

Page(pageObject);
```

- pages/simple.wxss

```
view {
  display: block;
  position: relative;
}

.main {
  padding: 0 20px;
}

text {
  display: block;
  margin-top: 20px;
}

.text {
  padding: 10px;
  padding-left: 6px;
}

.text.on {
  color: #fff;
  background: #009CD5;
}

.btn {
  position: absolute;
  left: 0;
```

```
  height: 40px;
}

.move {
  margin-top: 20px;
}

.move button {
  margin-top: 10px;
}
```

这个例子主要是一些事件的熟悉、dataset 的定义等。

服务端的数据

那么我们怎么和服务端交互呢？其实很简单，微信这边相当于一个客户端，我们建立两个项目，一个客户端、一个服务端提供服务。

首先我们启动一个服务端项目：wx_server 其中就一个文件，`users.json` 直接提供 JSON 数据，使用 `serve` 或者 `http-server` 启动一个服务。

- users.json
- [{
-   "name": "小撸"
- },{
-   "name": "骚龙龙"
- },{
-   "name": "菜胖娃"
- }]

然后其实剩下来都是客户端的事情了，先看下目录结构，我觉得 MVC 结构挺不错。

```
├── app.js
├── app.json
├── app.wxss
├── model
├── page
│   └── simple
├── service
└── util
    ├── ajax.js
    ├── assign.js
    ├── handler.js
    └── util.js
```

这里我建立 `model` 来做模型处理，`service` 做服务封装。然后 util 目录提供一些工具类。 直接上代码吧，这里的一个坑就是文件默认 `require` 模块不是 `.js` 结尾，必须写全，不知道正式版后会不会默认，所以最开始还是写严谨点好。

- assign 这个不贴了，就是 Object.assign 的一个封装。
- util/handler.js 这里提取项目所有的接口，这里我只是提取了一下服务 ROOT 前缀。
- util/ajax.js 主要是请求的简单封装，官方提供的是 `wx.request`，然后正式项目准备提取成 promise 的对象方式，这里先这样。

- handler.js

```
module.exports = {
    common: "http://localhost:3000/"
};
```

- ajax.js

```
var handler = require("./handler.js");
var assign = require("./assign.js");

var defaultConfig = {
    method: "GET",
    fail: function () { }
};

function ajax(config) {
    if(!config.url) {
        throw "必须传入请求的接口 url!";
    }
    if(!config.root) {
        config.url = handler.common + config.url;
    }

    var _config = assign(defaultConfig, config);
    console.log(_config);
    wx.request(_config);
}

module.exports = ajax;
```

- app.js

```
var ajax = require("./util/ajax.js");
var handler = require("./util/handler.js");
```

```
App({
  onLaunch: function () {
  },
  onShow: function () {
  },
  onHide: function () {
  },
  // 这里是关键的全局数据导出
  globalData: {
    ajax: ajax,
    handler: handler
  }
});
```

- pages/simple.wxml

```
<view class="page">
    <text class="page-text" wx:for="{{users}}">{{item.name}}</text>
</view>
```

- pages/simple.js

```
var app = getApp();
var globalData = app.globalData;
var ajax = globalData.ajax;
var handler = globalData.handler;

var pageObject = {
    data: {
        users: []
    },
    onLoad: function () {
        var self = this;
        ajax({
            url: "users.json",
```

```
            success: function (res) {
                var data = res.data || [];
                self.setData({
                    users: data
                });
            }
        });
    }
};

Page(pageObject);
```

- pages/simple.wxss

```
.page-text {
  display: block;
  line-height: 32px;
  font-size: 16px;
  color: #000;
  border-bottom: 1px solid #b2b2b2;
}
```

其实很简单，就是服务端提供接口即可，因为 APP 小程序这边可能有大小限制（目前可能是 1MB）所以应该是要分离的吧。

剩下还有一些生命周期，那个就看文档就应该 OK 了，匆匆写一篇入门文章