翻炒吧蛋滚饭：微信小程序初步入门知识梳理、收集

小程序 Demo

豆瓣电影

tabBar

- 文件：app.json
- 代码：

```
"tabBar": {
    "color": "#666",
    "selectedColor": "#268dcd",
    "boardStyle" : "white",
    "backgroundColor": "#fafafa",
    "list": [{
      "pagePath": "pages/douban/coming_soon/coming_soon",
      "iconPath": "image/coming",
      "selectedIconPath": "image/coming-active",
      "text": "即将上映"
```

```
    },
    {
      "pagePath": "pages/douban/in_theathers/in_theathers",
      "iconPath": "image/ing",
      "selectedIconPath": "image/ing-active",
      "text": "正在热映"
    }
    ]
  }
```

- 使用注意：

list 中的 pagePath，需要在 app.json 中添加后才能生效。

```
  "pages":[
      "pages/douban/coming_soon/coming_soon",
      "pages/douban/in_theathers/in_theathers"
  ],
```

Window

- 配置位置：app.json
- 用于设置小程序的状态栏、导航条、标题、窗口背景色。

数据绑定

page 中的 data 为页面的初始数据，初始化数据将作为页面的第一次渲染。data 将会以 JSON 的形式由逻辑层传至渲染层，所以其数据必须是可以转成 JSON 的格式：字符串，数字，布尔值，对象，数组。

渲染层可以通过 WXML 对数据进行绑定。格式：{{变量名}}，为 Mustache 语法。

```
// wxml 文件中
<view>{{text}}</view>
// js 文件中
Page({
  data:{
    text: 'hello world'
  }
})
```

以上页面的 view 标签内便会显示 hello world

条件渲染

- wx:if

在框架中，我们用 wx:if="{{condition}}"，来判断是否需要渲染该代码块。

- block

因为 wx:if 是一个控制属性，需要将它添加到一个标签上。但是如果我们想一次性判断多个组件标签，

我们可以使用一个 <block/> 标签将多个组件包装起来，并在上边使用 wx:if 控制属性。

setData

setData 函数用于将数据从逻辑层发送到视图层，同时改变对应的 this.data

的值。

注意：

直接修改 this.data 无效，无法改变页面的状态，还会造成数据不一致。

单次设置的数据不能超过 1024kB，请尽量避免一次设置过多的数据

```
Page({
  data:{
    text: 'hello world'
    textText: "asdasd"
```

```
  },
  viewTap:function() {
      this.setData({
          text: 'world Hello'
      })
  }
})
```

调用 viewTap 的时候，则可以刷新页面上引用该值的区域，同时给 text 赋值。

绑定点击事件

js 中自定义点击函数

```
viewTap:function() {
      console.log("点击了 view")
  }
```

wxml 中绑定点击事件

```
<text bindtap="viewTap">点我</text>
```

JS 中的打印

- 直接打印：

```
console.log("Hello World")
```

- 格式化打印：使用%来定义拼接类型，与 c 语言一致

```
var people = "Alex"
console.log("Hello %s", people)
```

- 拼接：

```
var name = "Bob";
console.log("The name is: " + name);
```

- 打印对象：

```
var people = {
    name: "Jack",
    age: 18
}
console.log("Jack：", people)
```

模块化

我们可以将一些公共的代码抽离成为一个单独的 js 文件，作为一个模块。模块只有通过

module.exports 或者 exports 才能对外暴露接口。

例：

```
module.exports = {
    someFunction:function() {
        console.log("打印了")
    }
}
```

调用

```
var functions = require('../functions.js')
functions.someFunction()
```