少年练剑：小程序学习笔记：HelloWorld，json 配置项，视图容器

一：HelloWorld

准备工作

官方文档：https://mp.weixin.qq.com/debug/wxadoc/dev/?t=1475052055990

开发工具：

https://mp.weixin.qq.com/debug/wxadoc/dev/devtools/download.html?t=1475052055364

开发工具安装好后，新建一个项目，记得取消勾选“创建 quick start 项目”

从 Hello，World！开始

了解下目录结构

小程序 = app 程序主体 + 页面 pages

app 程序主体文件结构：

1. app.js（必填）

   小程序的逻辑。通过 app()函数注册一个小程序，并指定生命周期函数。

2. app.json（必填）

   小程序的公共设置。页面路径、窗口表现、tabBar、网络超时等配置项。

3. app.wxss

   小程序公共样式表。可以理解成 common.css，不是必填，每个页面可以有自己的样式表，也可以引用公共样式表。

页面 page 文件结构：

1. js 文件（必填）

   页面逻辑。在这里注册页面，初始化数据、生命周期函数、事件处理函数等。

2. wxml 文件（必填）

   页面结构，可以理解成 html。

3. wxss

   页面样式表，理解成单个页面的 css 文件，可以为页面单独设置样式。

4. json

   页面配置。只能配置 window 相关配置项，改变窗体表现。

创建 app.js

通过开发工具的提示，使用 App 函数，快速生成一个小程序。

```
App({
  onLaunch: function () {
    console.log('App Launch')
  },
  onShow: function () {
    console.log('App Show')
  },
  onHide: function () {
    console.log('App Hide')
  }
})
```

创建 app.json

一个 app.json 文件的配置项包含：

1. pages（必填）---页面路径管理

   一个数组，每一项是字符串“路径+文件名”，指定小程序由哪些页面组成，数组第一项作为初始页面。文

   件名不需要后缀。

2. "pages":[
3. "pages/index/index"
4. "pages/logs/logs"

   ]

   后面这几个不是必备，等以后用到的时候再展开

5. window---设置窗口表现
6. tabBar---设置 tabBar
7. networkTimeout---设置网络请求超时时间
8. debug

所以我们的 app.json 代码如下

```
{
    "pages":[
        "pages/index/index"
```

```
    ]
}
```

创建 index 页面

为了便于管理，一般页面会放在 pages 文件夹下，一个页面一个子文件夹，里面包含 js 文件、wxml 文件、wxss 文件、json 文件（4 个文件名必须一致）。

index.js

用 Page 函数快速生成页面实例

```
Page({
  data:{
    // text:"这是一个页面"
  },
  onLoad:function(options){
```

```
    // 页面初始化  options 为页面跳转所带来的参数
  },
  onReady:function(){
    // 页面渲染完成
  },
  onShow:function(){
    // 页面显示
  },
  onHide:function(){
    // 页面隐藏
  },
  onUnload:function(){
    // 页面关闭
  }
})
```

index.wxml

把 view 换成 div 是不是很容易理解

```
<view id="hello-world">
    Hello,World!
<view>
```

index.wxss

样式表的使用基本跟 css 一样

```
#hello-world{
    text-align: center;
    margin-top: 75%;
}
```

编译运行


iPhone 6
wifi
Hello,World!

二：json 配置项

准备工作

14-46-45.jpg

1. 创建 my 页面（基础工作不在复述，参照上一篇）；
2. 创建图片文件夹 images，及子目录 tabBar；
3. 前往 http://iconfont.cn，下载任意两个 icon 及对应的高亮颜色版，60px，保存到 tabBar 目录下；

改变下导航栏

在 app.json 中添加 window 相关配置项：

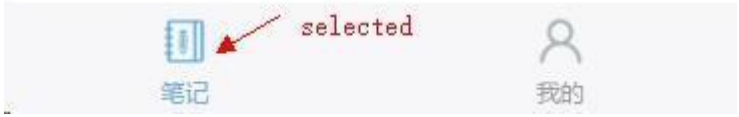
注意：json 中是不能加注释的，这里注释只是为了说明，请自行删除

```
"window":{
        "navigationBarTitleText":"小程序学习笔记",    //导航栏标题
        "navigationBarTextStyle":"black",  //标题颜色，默认 black，可选 white
        "navigationBarBackgroundColor":"#eee",  //导航栏背景色，默认#000

        //下面影响的是下拉刷新的部分，还没研究出怎么实现
        "backroundColor":"#555",      //窗口背景色，默认#fff
        "backroundTextStyle":"dark",    //下拉字体样式，默认 dark，可选 light
```

```
        "enablePullDownRefresh": true    //是否支持下拉刷新，默认 false
    }
```

写个 tabBar 吧



tabBar 最少 2 个，最多 5 个 tab

在 app.json 中添加 tabBar 配置项：

```
"tabBar":{
        "color":"#999",    //字体颜色，必填
        "selectedColor":"#56abe4", //选中状态字体颜色，必填
        "backroundColor":"#fff",    //背景色，必填
        "borderStyle":"white", //上边框颜色，非必填，默认 black，可选 white
        //list 数组，为每一个 tab 的配置，必填。tab 排序与数组排序一致
        "list":[
```

```
            {
                "pagePath":"pages/index/index",     //页面路径
                "iconPath":"images/tabBar/article.png",    //icon 路径
                "selectedIconPath":"images/tabBar/articleHL.png",    //选中状态 icon 路径
                "text":"笔记"    //文字
            },
            {
                "pagePath":"pages/my/my",
                "iconPath":"images/tabBar/my.png",
                "selectedIconPath":"images/tabBar/myHL.png",
                "text":"我的"
            }
        ]
    }
```

page 的 json

page 也是支持 json 文件的，不过只能配置 window 相关的，因此无需写 window 这个键

例如，在 my.json 添加下面代码，改变了标题栏文本为“我的”，而不是“小程序学习笔记”。

```
{
    "navigationBarTitleText":"我的"
}
```

json 配置项里还剩下 networkTimeOut 和 debug，以后用到再说吧

三：视图容器

准备工作

1. 新建 view 页面（4 个文件都包含，并修改导航栏标题文字为“视图容器”）；
2. 新建 app.wxss 公共样式表；
3. 修改 tabBar “我的”路径指向 view 页面；

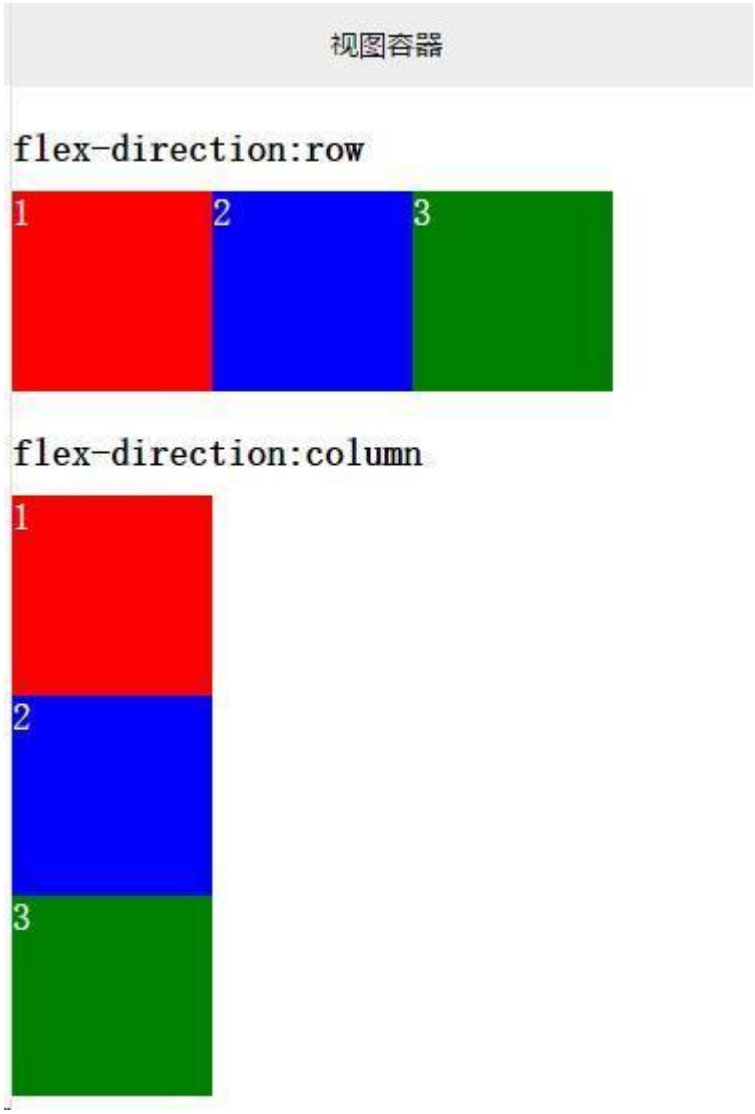
创建公共样式

因为后面都是学习各个组件的使用，页面结构基本保持一致，一个 view 包裹着标题+组件内容。因此这

里创建公共样式，代码如下：

```
.section{
    margin: 20px 0;
}
.section-title{
    margin-bottom: 10px;
    font-weight: bold;
}
```

view 视图容器的两种排列布局



行布局 row

<!--横向排列 view-->

```
<view class="section">
    <view class="section-title">flex-direction:row</view>
     /*通过 flex-direction：row  样式控制*/
    <view class="flex-wrp" style="flex-direction:row;">
        <view class="flex-item box-red">1</view>
        <view class="flex-item box-blue">2</view>
        <view class="flex-item box-green">3</view>

    </view>
</view>
```

列布局 column

```
<!--竖向排列 view-->
<view class="section">
    <view class="section-title">flex-direction:column</view>
    /*改变 row 为 column*/
    <view class="flex-wrp" style="flex-direction:column;">
```

```
        <view class="flex-item box-red">1</view>
        <view class="flex-item box-blue">2</view>
        <view class="flex-item box-green">3</view>
    </view>
</view>
```

样式

box-red、box-blue、box-green 的背景颜色大家自己补充一下，这里就不在写出来了。

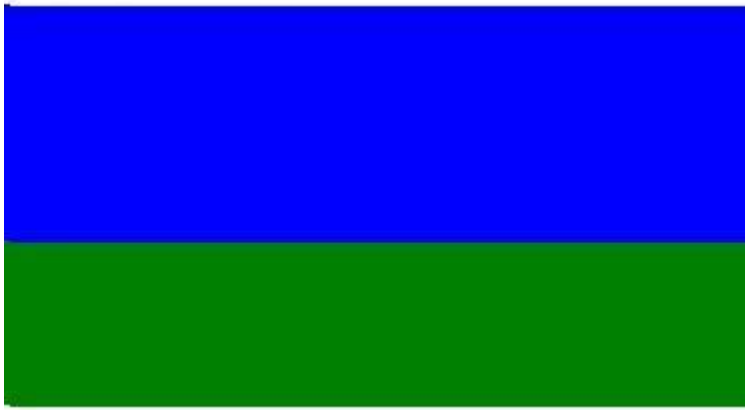
```
.flex-item{
    width: 100px;
    height: 100px;
    color: #fff;
}
.flex-wrp{
    display: flex;   /*不要忘记*/
}
```

动手开始做才发现自己的 css 水平简直不过关，布局部分好多都不知道，如果有类似情况的同学，推荐

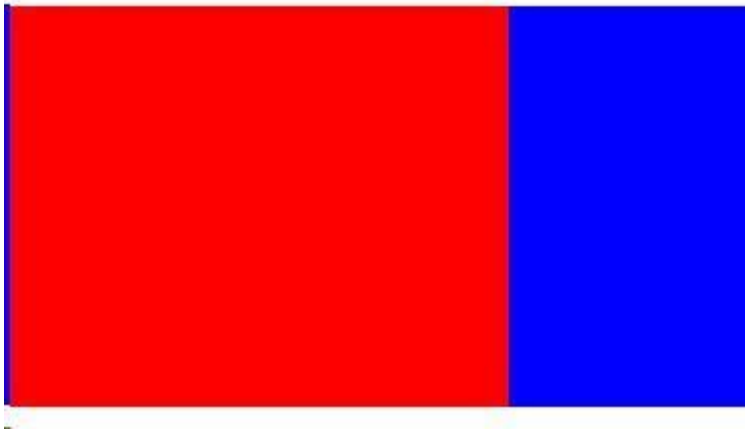
大家一个网站 http://zh.learnlayout.com/，共勉吧！我先慢慢磨完小程序。

支持滚动 scroll-view

scroll-view(-y)



scroll-view(-x)



竖向滚动

注意：为了让代码更加简洁，往后文章里我去掉了标题和外包的 view

```
<!--需要给定容器一个固定高度-->
<scroll-view scroll-y="true"    style="height:200px;">
        <view id="red" class="scroll-view-item box-red"></view>
        <view id="blue" class="scroll-view-item box-blue"></view>
        <view id="green" class="scroll-view-item box-green"></view>
</scroll-view>
```

样式

```
.scroll-view-item{
    width: 100%;
    height: 200px;
}
```

横向滚动

```
<!--记得加 white-space:nowrap;  规定不换行-->
<scroll-view scroll-x="true"     style="white-space:nowrap;">
```

```
        <view id="red-H" class="scroll-view-item-H box-red"></view>
        <view id="blue-H" class="scroll-view-item-H box-blue"></view>
        <view id="green-H" class="scroll-view-item-H box-green"></view>
    </scroll-view>
```

样式

```
.scroll-view-item-H{
    display: inline-block;   /*不要忘记*/
    width: 100%;
    height: 200px;
}
```

其他属性

scroll-top="100" ----> 设置竖向滚动条位置

scroll-left="100" ----> 设置横向滚动条位置

scroll-into-view="blue" ----> 值为某子元素 id，则默认滚动到该元素顶部

另外，还做了些有趣的尝试

- scroll-x 和 scroll-y 同时为 true，支持双向滚动（需要内容宽高都超过 scroll 容器）；

- scroll-top 和 scroll-into-view 同时存在时，scroll-top 无效，不管数值多少；
- scroll-into-view 对横向滚动无效（不知道是不是我方法的问题）

其他事件属性暂时不展开，请自行查阅官方文档

轮播 swiper


```
<swiper indicator-dots="true" autoplay="true" current="1" interval="4000" duration="1000"
vertical="true" style="height:200px;">
        <swiper-item class="box-red"></swiper-item>
        <swiper-item class="box-blue"></swiper-item>
        <swiper-item class="box-green"></swiper-item>
```

```
    </swiper>
```

swiper-item 宽高固定为 swiper 的 100%，相当于自动填充

属性说明

indicator-dots="true" ----> 是否显示面板指示点（默认值 false）

autoplay="true" ----> 是否自动切换（默认 false）

current="1" ----> 起始位置（默认 0，即第一张）

interval="4000" ----> 自动切换时间间隔（默认 5000，单位毫秒，即 5 秒）

duration="1000" ----> 滑动动画时常（默认 1000，单位毫秒）

vertical="true" ----> 竖向轮播（默认 false）

遗留问题

发现默认值是 false 的属性，如果我们写出来，例如 vertical="false"，实际仍然按照 true 执行，结果是轮播竖向显示。其他的属性我也尝试了，包括上面的 scroll-y="false"，结果仍然支持竖向滚动，大家可以试一下。

但是看微信官方 demo 里的代码是可以的，通过事件改变属性，false 就是 false，实在不清楚，先记在这里吧。

---------已解决-----------

学习下一节基础内容时，得到启发，google 一下，解决了问题。

直接写在组件属性里，如 vertical="false"，实际上 false 被当成了一个字符串来处理，非布尔值。而字符串不为空的情况下，布尔值=true，所以发生了上面遇到的情况。

附：javascript 表达式为假的情况，除此之外其他都为真

false、0、字符串的空''、NaN、undefined、null

因此 vertical="false"我们可以这样实现：

1. vertical=""
2. 通过 js 文件 data 初始化数据，vertical：false 或者 任何一个为假的表达式
3. 不写，如果默认值就是 false 的话（推荐）

另外也发现，对于这些默认值 false 的属性，如果需要改成 true，直接加上去就可以，例如这样：

```
<!--直接写 scroll-y，而不是 scroll-y="true"-->
<scroll-view scroll-y style="height:200px;">
        <view id="red" class="scroll-view-item box-red"></view>
        <view id="blue" class="scroll-view-item box-blue"></view>
        <view id="green" class="scroll-view-item box-green"></view>
```

</scroll-view>

至于为什么，暂时还没明白。

总结一下：对于默认值 false 的属性，如果想要改成 true，直接加上去即可，如果不需要，去掉就行