

## 第一章

# 小程序到底是什么？

顾名思义，「小程序」本质上与我们平常经常使用的 App 和操作系统一样，都是一段电脑程序。你可以将小程序理解为「运行在微信上的 app」。

与普通的 App 不同的是，小程序的语言使用网页前端的技术栈，并且使用 JavaScript 运行环境，相当于是一个浏览器。

但是，小程序也并非是单纯的 HTML5。与普通的 HTML5 不同的是，小程序具有高级的硬件处理接口（例如蓝牙、重力感应等），同时运行环境也更接近原生应用，效率更高。

正因体积小，小程序才可以做到「即开即用」——对于用户感知来说，小程序几乎是点击后等待一两秒，就可以使用，就像是打开网页链接一样，使用后，用户甚至不需要额外管理小程序。

除了使用层面本身的良好体验，由因为小程序是直接集成于微信中的，所以它还可以配合微信完成传统 App 和 HTML 5 无法达成的功能。

例如，分享卡片可以携带相应微信群信息，当用户从微信群进入小程序，开发者可以了解到用户从哪一个群进入小程序。

## 小程序是如何发展起来的？

在 2016 年 9 月，有部分开发者收到了微信的邀请，尝试新的「小程序」平台，引发大量关注。

在此前，张小龙曾在公开场合宣布，微信将会推出「应用号」平台。外界普遍猜测，「小程序」即此前张小龙所提到的「应用号」。

直到 2016 年 11 月，小程序平台宣布公测，所有符合资格的组织都可以注册小程序帐户。此次公测正式引爆公众对小程序的热情，许多企业连夜注册小程序帐户，希望可以尽快尝试这个新平台。

2017 年 1 月 9 日，小程序正式开放使用。截止今日，市面上已有不计其数的微信小程序。微信用户也已经逐渐习惯使用小程序，小程序变为他们的「微信生活」中，不可或缺的一部分。

## 一、小程序有什么特点？

在前面，我们提到：小程序既不是网页，也不是 App，它是一种融合了网页和 App 两者特点的新应用形态。

相比较于 App 和网页，小程序具体拥有以下特点：

- 随时加载，随时使用
- 拥有强大接口和能力
- 易学、易开发

## 二、了解小程序开发语言

今天的教程，我们将不讨论有关小程序开发的具体内容，我们将会为大家带来更基础的一些东西，帮助大家在未来快速上手开发小程序。

### 1. 视图描述语言

你在小程序中看到的文字、图片、按钮等，都被称做「视觉组件」。

在小程序中，想要控制这些视觉组件，我们需要使用到 WXML、WXSS 两种语言。如果你不了解它们，你可能会认为「微信创造了新的编程语言」。

但实际上，WXML 和 WXSS 都是从以往就有的语言演变的。

## 1) WXML

WXML 的全称是微信标记语言（WeiXin Marked Language），从名字中我们就知道，它其实是由 XML 和 HTML 演变而来的。

WXML 的作用是描述小程序页面中应该有什么视觉元素。它的语法并不复杂。最简单的 WXML 代码如下所示：

```
<text>Hello World</text>
```

它的意义是，在小程序页面中显示 Hello World 的文字（text）。

## 2) WXSS

WXSS 的全称是微信样式表（WeiXin Style Sheet），它的语法与 CSS 没有什么区别。

它的作用是，定义页面中的元素的样式是怎样的。例如：

```
text{  
    color: red;  
}
```

它的意思是，将 WXML 中 <text> 元素的字体颜色（color）修改为红色（red）。

## 3) WXML 与 WXSS 协同使用

将 WXML 和 WXSS 放在一起用，我们就可以自由控制小程序中的视觉元素展示方式、样式了。

如果你有一组元素需要使用同样的样式，或是有一个特定的元素需要使用某个格式，那么你可以使用类（Class）和 ID 特性。

使用类，你可以针对一组同类视觉元素，修改样式。例如，我们希望将小程序中的所有用户名都显示为红色，我们可以这样写：

```
<!-- WXML -->  
<text>知晓程序</text>  
  
/* WXSS */  
.username{  
    color: red;  
}
```

## 2. 逻辑语言

通过视图层，用户就可以与小程序产生交互。但小程序的逻辑处理（包括网络数据交互、运算、逻辑判断等），都需要用 **JavaScript (JS)** 语言写成的代码完成。

我们先从页面逻辑入手，编写简单的 **Hello World** 程序来接触 **JS** 代码。

```
Page({
  onLaunch() {
    console.log('Hello World!')
  }
})
```

运行它，我们就可以在控制台中看到结果了。

只需简单几行代码，就是一个完整的小程序 **JS** 程序。调用 **JS** 函数很简单，只需要写函数的名字，并在后面括号中附带参数，就可以调用。

函数名(参数 1, 参数 2, ...)

有关于小程序开发的三种语言 **WXML**、**WXSS** 和 **JS** 的简单介绍到这里。

请注意，本篇只是对这些语言语法的基础讲解。虽然本教程会在未来的课程中教授更深层次的内容，但同时我们也建议，没有前端开发基础的同学，可以先去学习一些基础的前端开发。

## 开发前准备

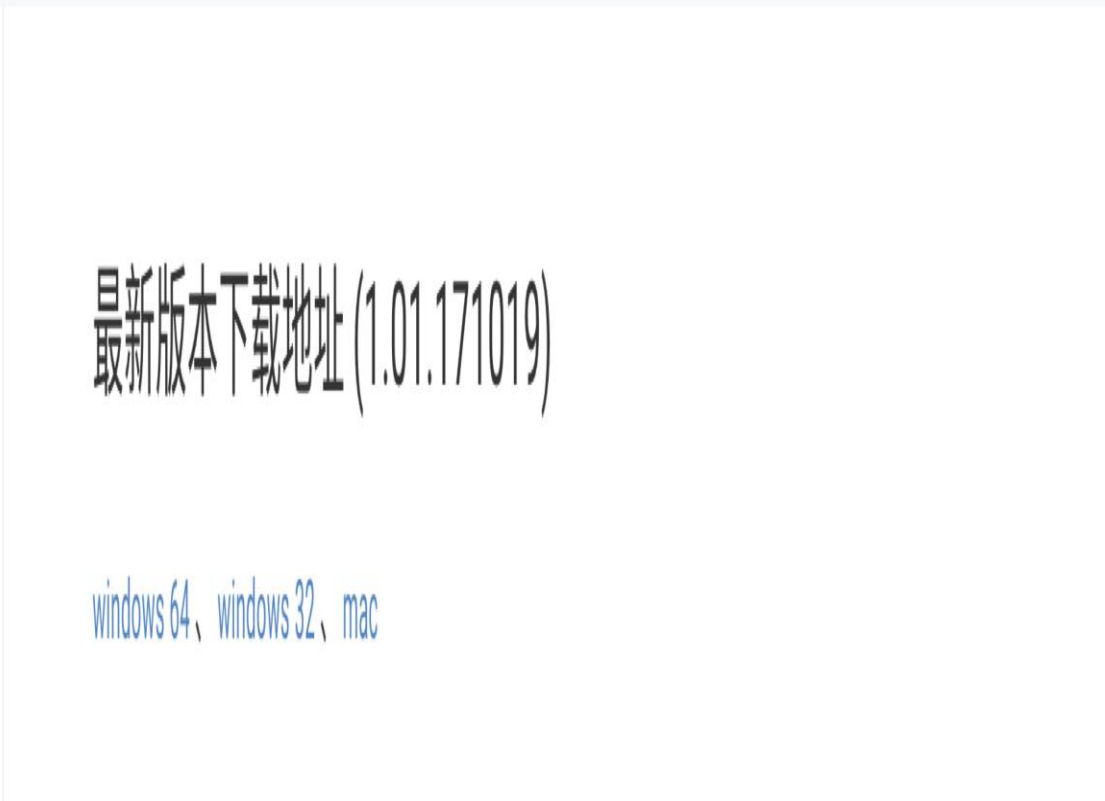
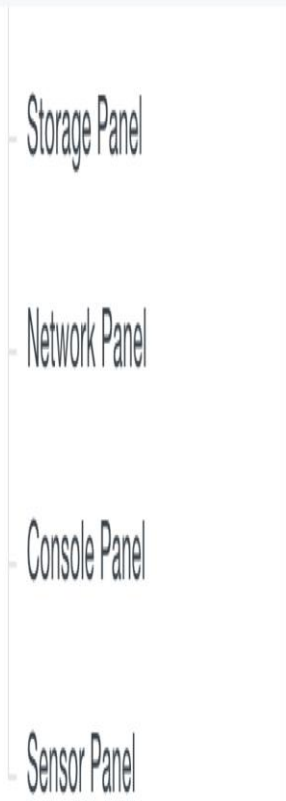
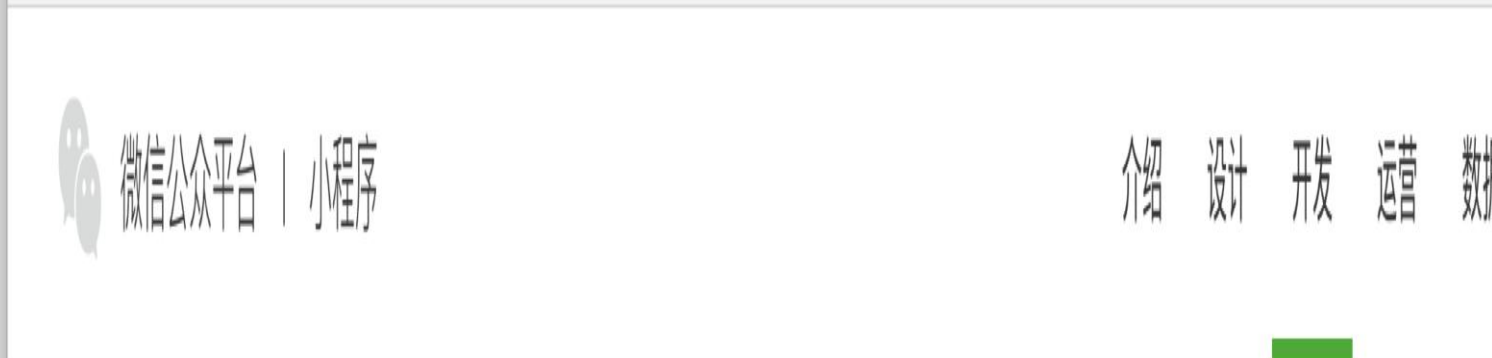
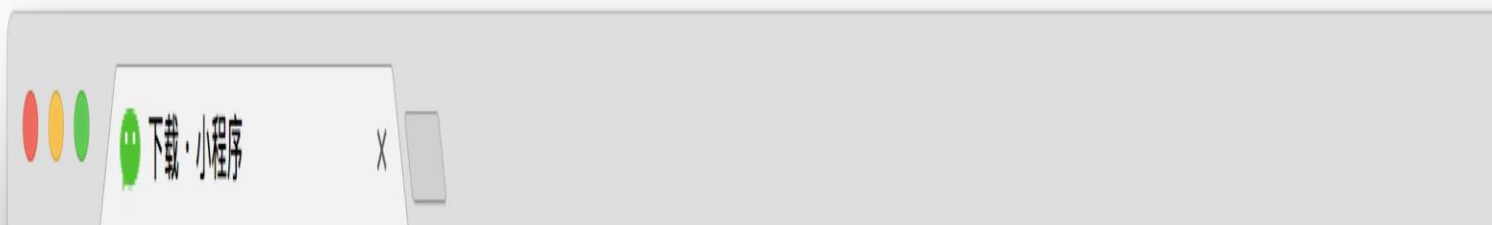
工欲善其事，必先利其器。在正式开发小程序之前，我们还需要做一些准备工作。

### 一、开发者工具

首先，我们就来初步认识小程序的开发环境和必备软件：「微信 **Web** 开发者工具」。

## 二、下载、安装开发者工具

我们可以到 [mp.weixin.qq.com](http://mp.weixin.qq.com) 下载这个工具。由于官方文档不断在变化，所以地址随时有可能会变更。微信关注「知晓程序」公众号，回复「工具」，就可以随时获取最新的开发者工具下载地址。



下载好开发者工具之后，我们自然需要进行安装。**macOS** 与 **Windows** 两个版本的开发者工具安装方式有些许差异：

- **Windows** 版下载好安装程序后，直接双击打开进入安装流程。安装完毕后，一般可在桌面即可启动开发者工具。

- **macOS** 版下载好镜像文件后，直接拖动到「**Application**（应用程序）」文件夹中，在 **LaunchPad** 就可以启动开发者工具。  
大家可以根据相应提示，安装、启动开发者工具。

### 三、新建小程序项目

安装完毕后，我们就可以在开发者工具中，新建小程序工程项目了。

如果是首次启动「微信 **Web** 开发者工具」，你需要使用微信号扫描二维码进行登录。在开发过程中，登录的微信号将会用于微信开放能力相关接口（例如，获取用户资料、发送模板消息等）的调试。

登录完成后，我们就进入项目类别选择的窗口。在这一步，我们选择「小程序项目」。



首次点击「小程序项目」时，开发者工具会直接要求我们新建一个小程序项目。

- 项目目录：在这个字段中为小程序工程选择一个文件夹，用于存放小程序项目的代码。

- AppID：如果你已经有一个小程序的 AppID，可以在这里将它填入。如果没有，可以选择下面的「点此体验」链接。

- 项目名称：为小程序项目起一个名字。这个名字只针对开发者工具中识别，不会影响小程序本身对外显示的名字。

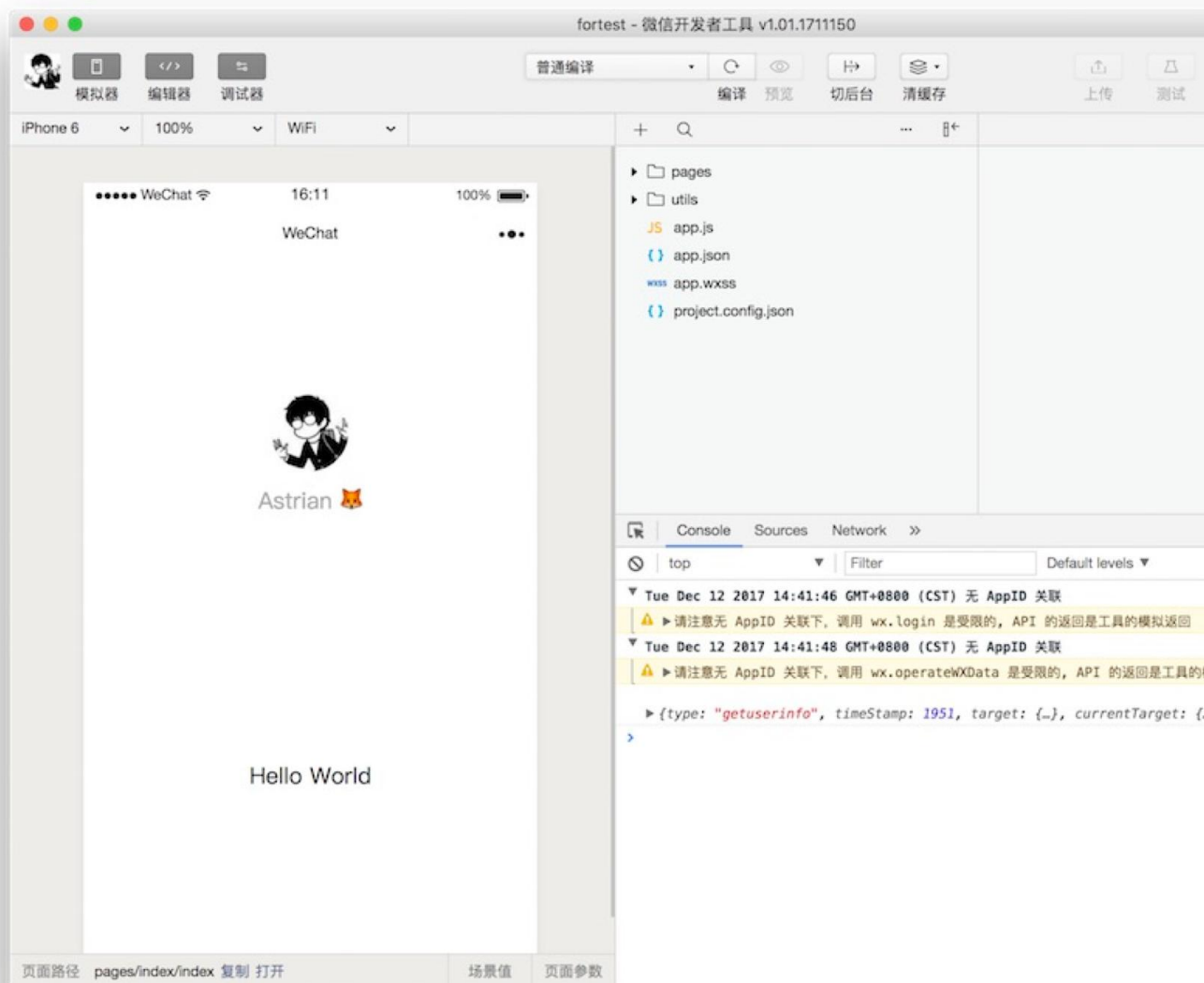
确认无误，点击「确定」，就可以新建小程序项目。之后，我们就进入「微信 Web 开发者工具」的主界面。

## 使用开发者工具进行开发



「微信 Web 开发者工具」主要包含四个部分。

- 顶部为工具栏，可对开发者工具的帐户、窗口显示进行调整，或是执行编译、预览或上传小程序的操作。
- 左侧为模拟器，开发中的小程序代码，可以直接在预览区中查看模拟运行效果。
- 右侧上半部分为编辑器，可以在其中浏览小程序工程目录、直接编辑小程序代码。
- 右侧下半部分为调试器，小程序的运行结果、输出等信息，会在这部分进行显示。



开发者工具提供的功能、界面等都相对清晰、简单，我们只需要在代码编辑器写好代码，在模拟器就可以看到相应运行效果；如果运行出错，在调试器中，也能找到详细的错误信息。

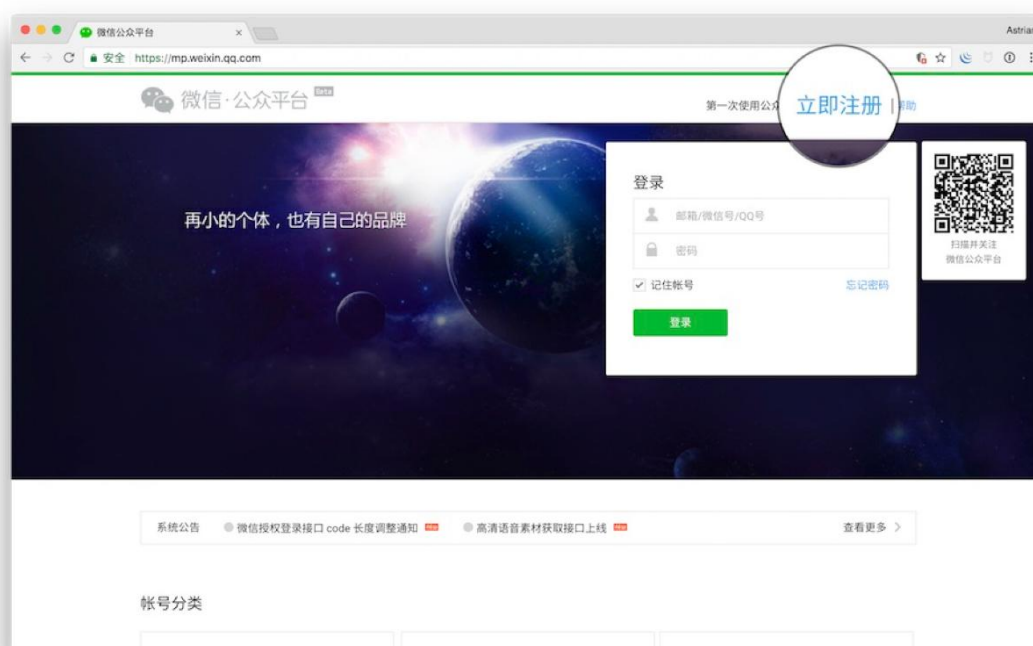
## 四、小程序账号

我们已经了解如何使用开发者工具来开发小程序、管理小程序项目。此时，如果你还没有注册小程序，你只能在「预览」状态下新建小程序项目，一部分开发功能将无法使用。

那我们应该如何注册小程序帐户呢？注册的时候，需要留意哪些问题呢？在本文中，我们就来手把手教大家，如何注册一个小程序。

小程序注册流程，与公众号注册流程差不多。

你需要进入 [mp.weixin.qq.com](https://mp.weixin.qq.com)，点击右上角的「立即注册」链接，在「帐户类型」页面中，选择「小程序」。



点击「小程序」后，你需要填写邮箱、密码等帐户基本信息。它们用于你注册后登录小程序。

需要注意的是，已经绑定其他的公众号、小程序、个人号的邮箱，不能重新注册新的小程序。

也就是说，你在微信里与微信号绑定的邮箱，或是在公众平台注册公众号所用的邮箱，是不能用于注册小程序的。

 微信公众平台 | 小程序

## 小程序注册

① 帐号信息 — ② 邮箱激活 — ③ 信息登记

每个邮箱仅能申请一个小程序

已有微信小程序？[立即登录](#)

邮箱

作为登录帐号，请填写未被微信公众平台注册，未被微信开放平台注册，未被个人微信号绑定的邮箱

密码

字母、数字或者英文符号，最短 8 位，区分大小写

确认密码

请再次输入密码

验证码



[换一张](#)

☐ 你已阅读并同意《[微信公众平台服务协议](#)》及《[微信小程序平台服务条款](#)》

注册

提交后，填写的邮箱会收到一封确认注册邮件，你需要点击邮件中的确认链接，然后填写并验证小程序的主体信息。

填写主体信息的过程中，不同的主体，验证方式也有差异。

如果你是以个人身份进行注册，在「主体类型」一项，你需要点击「个人」。之后，你只需要在弹出的表单中填写资料、完成验证，就可以直接完成注册小程序。

需要注意的是，验证过程中，你需要验证你的手机号，并使用你自己的微信号扫码确认。

微信公众平台 | 小程序

文档社区

小程序注册

① 帐号信息

② 邮箱激活

③ 信息登记

用户信息登记

微信公众平台致力于打造真实、合法、有效的互联网平台。为了更好的保障你和广大微信用户的合法权益，请你认真填写以下登记信息。  
为表述方便，本服务中，“用户”也称为“开发者”或“你”。

用户信息登记审核通过后：  
1. 你可以依法享有本微信公众平台帐号所产生的权利和收益；  
2. 你将对本微信公众平台帐号的所有行为承担全部责任；  
3. 你的注册信息将在法律允许的范围内向微信用户展示；  
4. 人民法院、检察院、公安机关等有权机关可向腾讯依法调取你的注册信息等。

请确认你的微信公众平台主体类型属于政府、媒体、企业、其他组织、个人，并严格按照对应的类别进行信息登记。  
点击查看微信公众平台信息登记指引。

主体类型

如何选择主体类型？

个人

企业

政府

媒体

其他组织

个人类型包括：由自然人注册和运营的公众帐号。  
帐号能力：个人类型暂不支持微信认证、微信支付及高级接口能力。

主体信息登记

身份证姓名

信息审核成功后身份证姓名不可修改；如果名字包含分隔号“”，请勿省略。

身份证号码

请输入您的身份证号码。一个身份证号码只能注册 5 个公众帐号。

管理员手机号码

1.2秒后可重发

请输入您的手机号码。一个手机号码只能注册 5 个小程序。

短信验证码

319628

无法接收验证码？

请输入手机短信收到的 6 位验证码

管理员身份验证

身份验证成功  
作为小程序的管理员。

继续

关于腾讯 | 文档中心 | 帮助中心 | 客服中心 | 联系邮箱

Copyright © 2012-2017 Tencent. All Rights Reserved.

以组织身份注册时，你首先需要根据实际情况，在「主体类型」中选择适合的组  
织类型。之后，页面会根据你的选择，显示信息登记的表单。

主体类型

如何选择主体类型？

个人	企业	政府	媒体	其他组织
----	----	----	----	------

企业包括：企业、分支机构、个体工商户、企业相关品牌。

## 主体信息登记

企业类型

☒ 企业 ☐ 个体工商户

企业包括：企业、分支机构、企业相关品牌等

企业名称

知晓程序

需与当地政府颁发的商业许可证书或企业注册证上的企业名称完全一致，信息审核审核成功后，企业名称不可修改。

营业执照注册号

请输入15位营业执照注册号或18位的统一社会信用代码

注册方式

☐ 向腾讯公司小额打款验证

☐ 微信认证

① 填写企业对公账户

为验证真实性，此对公账户需给腾讯打款验证。注册最后一步可查看打款信息，请尽快联系贵公司/单位财务进行打款。

② 注册最后一步，需用该对公账户向腾讯公司进行打款

③ 腾讯公司收到汇款后，会将注册结果发至管理员微信、公众平台内信

④ 打款将原路退回至您的对公账户

微信注册并认证，无需小额打款验证，需支付300元审核费用。提交认证后会在1-5个工作日完成审核。在认证完成前小程序部分能力暂不支持。[查看详情](#)

## 管理员信息登记

管理员身份证姓名

请填写该小程序管理员的姓名，如果名字包含分隔号“-”，请勿省略。

管理员身份证号码

请输入管理员的身份证号码，一个身份证号码只能注册5个小程序。

管理员手机号码

获取验证码

请输入您的手机号码，一个手机号码只能注册5个小程序。

短信验证码

无法接收验证码？

请输入手机短信收到的6位验证码

管理员身份验证

请先填写企业名称与管理员身份信息

继续

对于企业来说，小程序可以选择不经过认证而完成注册。填入企业名称之后，页面会显示可用的注册方式，包括小额打款以及微信认证。

但是，未认证的小程序在开发者数量、可用能力方面，会与已通过认证的小程序会有一些差异。

在注册过程中，页面也会要求填写小程序管理员的信息。

与个人主体小程序注册流程一致，你同样需要提供自己的个人信息，也需要进行短信验证、扫码验证。注册完毕后，你就可以对小程序进行管理。

主体验证通过后，你就可以进入到小程序的后台。之后，你可以根据页面提示，补充小程序的名称、标志，然后下载开发者工具，开始开发你的小程序。

## 第二章

了解小程序开发必备的基础知识之后，我们就要开始自己动手，制作我们自己的第一个小程序。

这个小程序与开发者工具默认新建的初始工程类似，会显示用户的头像和名字，同时会向用户显示问候语。

## 准备开发

在开始开发之前，我们需要完成一系列准备工作。

### 在开发者工具中新建工程

在前一章中，我们已经谈到如何在开发者工具中新建工程。我们只需按照实际情况，在开发者工具中新建项目即可。

注意，我们需要在项目目录中选择一个完全空白的目录（最好直接新建），并取消最后的「建立普通快速启动模板」勾选框。这样，我们就能新建空白的小程序项目。

我们会在本章一步步带领大家一同完全从零开始创建小程序文件。

← 小程序项目管理



## 小程序项目

编辑、调试小程序

项目目录

/Users/[redacted]/welcome



AppID



若无 Appid 可 [注册](#) 或体验: [小程序](#) / [小游戏](#)

项目名称

welcome



建立普通快速启动模板

确定



接下来，我们就进入开发者工具中，新建几个小程序必要的文件。

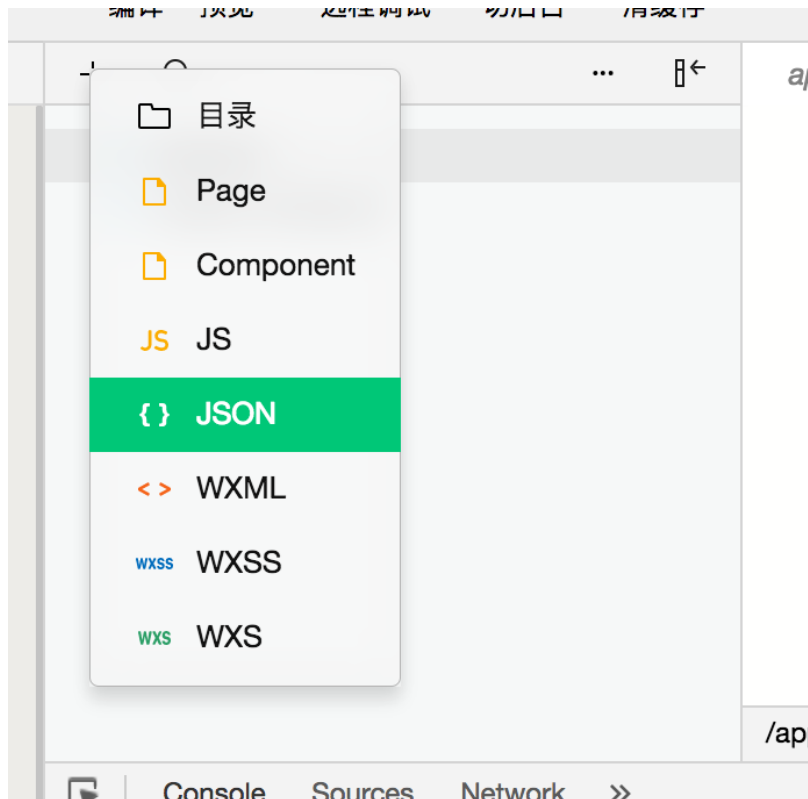
## 调整配置文件

熟悉 iOS 或 Android 开发的开发者都知道，App 开发会有一个名叫「清单文件」的东西，它记录了 app 的图标、名称、兼容性等 pp 的元信息，帮助系统更好地运行这个 App。

而在微信小程序中，也有功能类似的文件，就是 `app.json`。它记录小程序一些基础配置：

- 小程序的所有页面路径
- 小程序全局界面表现
- Tab 标签页配置
- 网络超时
- 调试模式开启或关闭
- 

我们可以这样新建一个 `app.json` 文件：在开发者工具的编辑器（通常在右上角）中，点击左上方的「+」号，选择 JSON，并命名为 `app.json`。



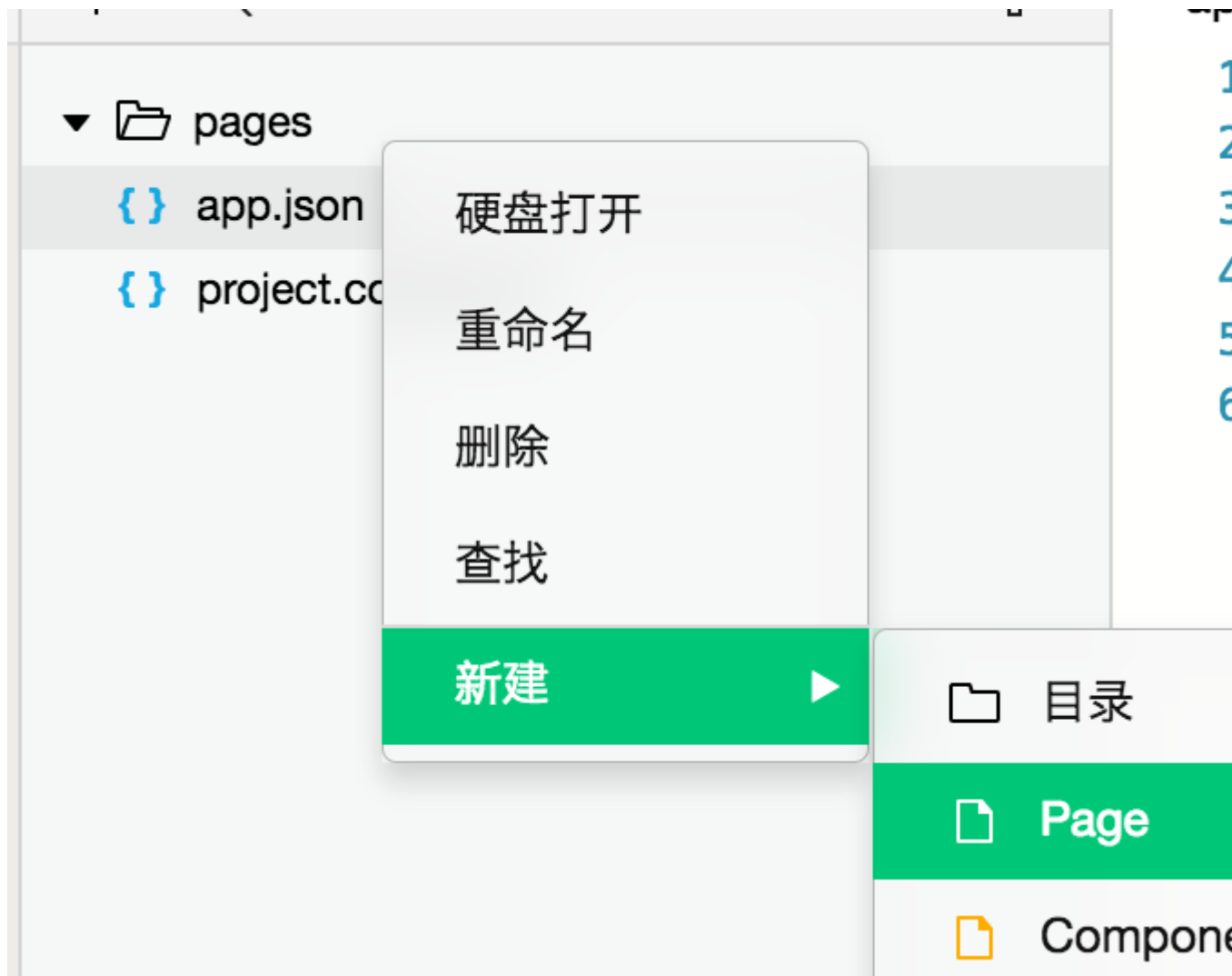
新建文件后，我们还得在 `app.json` 写入配置，才能让小程序正常工作。在此，我们可以直接输入以下代码直接保存。

```
{}
```

保存后，我们可以看到编辑器依然在报错。这是因为 `app.json` 目前是空的。

## 小程序页面路径配置

接下来，在编辑器中新建「`pages`」文件夹，并对着文件夹点击右键，新建「`Page`」并起名为 `welcome`。



保存后，`pages` 文件夹会多出几个文件，而 `app.json` 也有相应变化：

```
{
  "pages": [
    "pages/welcome"
  ]
}
```

可以看到，开发者工具已经自动地将我们在 `pages` 文件夹中新建的 `welcome` 页面路径，放入到 `pages` 数组中。

在 `app.json` 中，`pages` 数组是必填的。它规定小程序中所有页面的地址，同时规定了小程序启动时的首页，就是 `pages` 数组的首位所指的页面。

当 `pages` 成功配置后，小程序就可以正常运行了（新建页面文件后，小程序已经正常运行无报错了）。但如果想要一些个性化配置，我们依然可以继续修改 `app.json`。

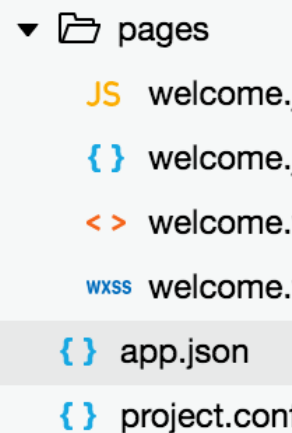
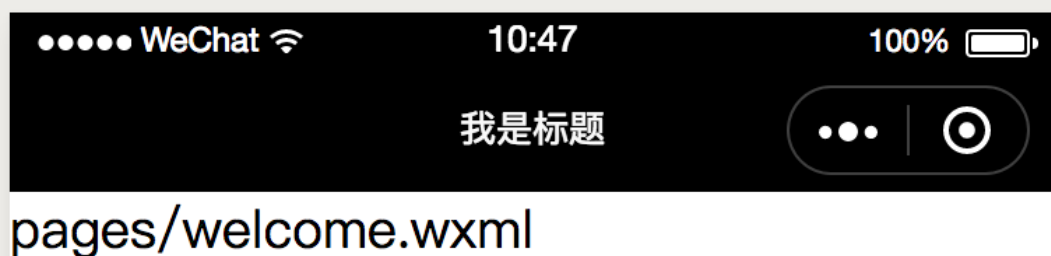
## 修改小程序页面标题

我们以修改小程序页面的全局页面标题为例，了解 `app.json` 更多个性化设置。

在 `app.json` 中，全局页面标题项目被分配在 `window` 大项中，名为 `navigationBarTitleText`。修改后的 `app.json` 形如这样：

```
{
  "pages": [
    "pages/welcome"
  ],
  "window": {
    "navigationBarTitleText" : "我是标题"
  }
}
```

保存后，我们就可以实时看到小程序已经应用修改。



除了页面标题文字，类似标题栏颜色、Tab 栏内容等信息，都可以直接在 `app.json` 进行配置。

现在，我们的 `app.json` 已经配置完毕。接下来，我们就来为小程序增加几个视觉元素。

## 了解 WXML

### 插入文字

我们打开 `welcome` 文件夹下的 `welcome.wxml` 文件，可以看到开发者工具在自动生成页面的时候，默认生成的代码。

```
<!--pages/welcome.wxml-->
<text>pages/welcome.wxml</text>
```

没错，我们在预览区看到的「`welcome.wxml`」语句，便是由它进行控制的。

第一行的 `<!--welcome.wxml-->`，是一行注释。计算机在编译代码的时候，会跳过注释，不会对注释代码进行编译。

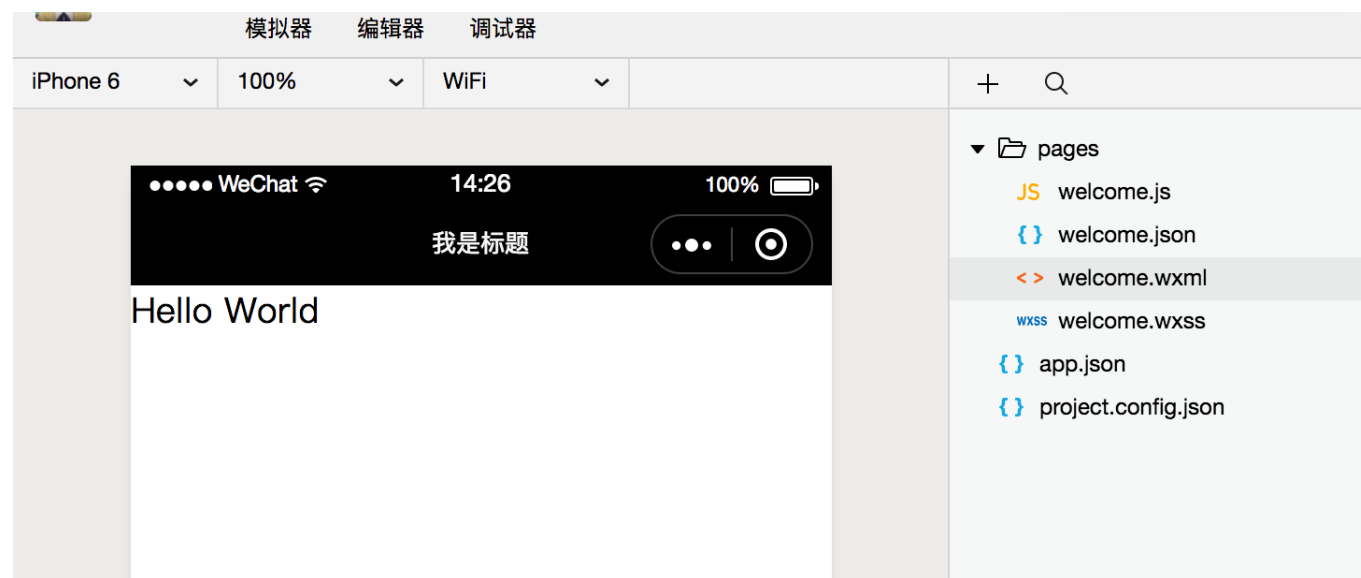
使用注释，我们可以对代码的功能，进行一些说明，以便未来对代码进行维护。

我们看一下第二行的代码。我们看到，这句代码的前后都是由尖括号包裹起来的部分，中间则是 `welcome.wxml`。

我们试图将中间的 `welcome.wxml` 修改成 `Hello World`（当然，也可以改成你自己喜欢的短语）。修改后的代码变成了这样：

```
<!--pages/welcome.wxml-->
<text>Hello World</text>
```

当我们保存、编译后，我们可以看到，预览区域中原先写着「welcome.wxml」的地方，已经变成了「Hello World」。



在小程序中，`<text></text>` 代表文字视觉组件。在它们中间插入的内容，将会直接显示在小程序的相应位置中。

## 插入图片

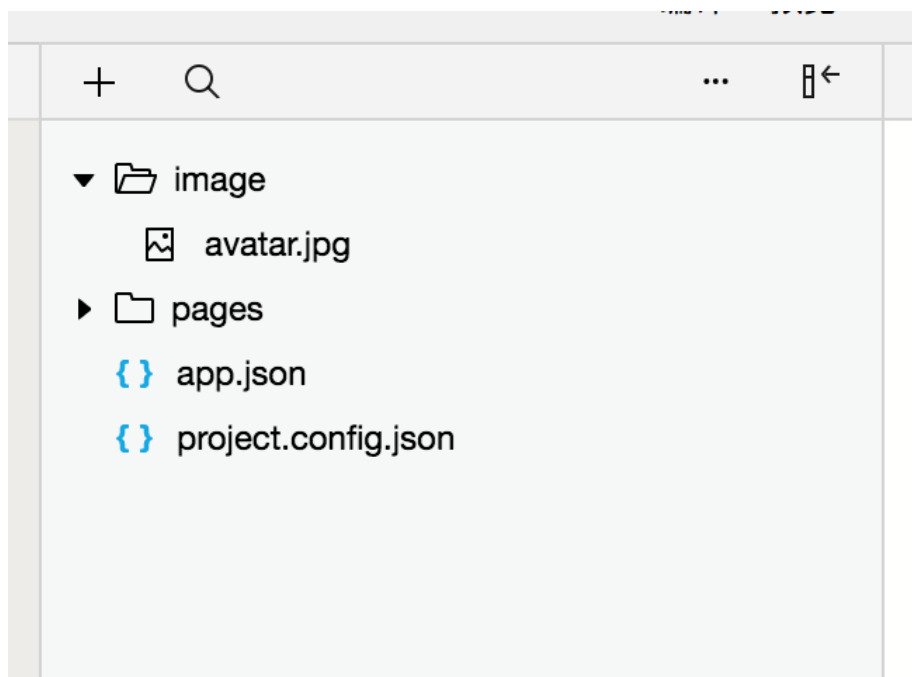
接下来，我们试图在我们的小程序里插入一张图片。

首先，我们在根目录下创建一个 **image** 的文件夹。然后，我们点击我们刚才新建的 **image** 文件夹，点击右边的「...」按钮，选择「硬盘打开」。



这时，电脑会打开我们的工程文件夹。我们在文件夹里，打开我们刚才新建的 `image` 文件夹，随意将一张图片放进文件夹里面。

接着，开发者工具会自动探测到文件的变化，然后重新刷新工程目录。刷新之后，我们就可以在 `image` 文件夹下，看到我们刚才放进去的文件。



之后，我们回到 `welcome.wxml`，插入刚才放入目录中的图片。

假设我们放入 `image` 目录的图片文件名称叫 `avatar.jpg`，那么我们需要在文件的底部插入这段代码：

```
<image src='/image/avatar.jpg'></image>
```

插入这段代码之后，`welcome.wxml` 文件看上去应该是这样的：

```
<!--pages/welcome.wxml-->
<text>Hello World</text>
<image src='/image/image.jpg'></image>
```

保存后，我们就可以在模拟器中，看到整体效果了。

与 `<text>` 一样，`<image>` 也是小程序的一个视觉组件，它代表在小程序里插入一张图片。

有所不同的是，`<image>` 组件内不应该插入任何内容。我们应该在 `<image>` 组件的 `src` 属性中，指定需要插入在小程序里的图片。

将代码保存，开发者工具会自动刷新预览。在预览中，我们可以看到图片虽然已经成功插入，但是比例似乎不太正确。





这是因为 `<image>` 本身拥有几种缩放、裁剪模式和尺寸大小，而默认尺寸为 `300x225 px`，默认模式为「完整填充」。

在 `<image>` 组件中,增加 `mode`,定义值为 `widthFix`,它可以让图片比例正常,并根据设定的宽度,自动按比例调整 `<image>` 元素的高度。

```
<image src='/image/avatar.jpg' mode='widthFix'></image>
```

保存后,可以看到小程序的图片显示就正常了。



## 给元素加「框」

在正式开发过程中，我们会将屏幕元素分隔成不同部分，每个部分可以套用相应样式，使用独立的样式代码，以提高编码效率。

例如，在传统 **HTML** 开发中，我们会利用 `<div>` 框元素对不同部分进行区分。而在小程序中，我们可以利用 `<view>` 达成同样的效果。它的使用方法与 `<div>` 几乎没有什么不同

在这里，我们会将其中的文字、图片元素「绑定」起来，作为屏幕的组成部分。

```
<!--pages/welcome.wxml-->
<view>
  <text>Hello World</text>
  <image src='/image/avatar.jpg' mode='widthFix'></image>
</view>
```

这样做，我们就可以统一处理文字和图片的样式和其他操作。

在下一章中，我们将会对屏幕上的这些元素做一些「美化」操作，并将它们换成用户的资料。

## 第三章手把手，动手编写一个简单小程序 (下)

### 界面美化与样式调整

在上一节，我们已经初步了解在小程序中增加视觉元素的方法。而在这一节，我们将会学习编写小程序的样式表，为我们的欢迎小程序「美化」一下。

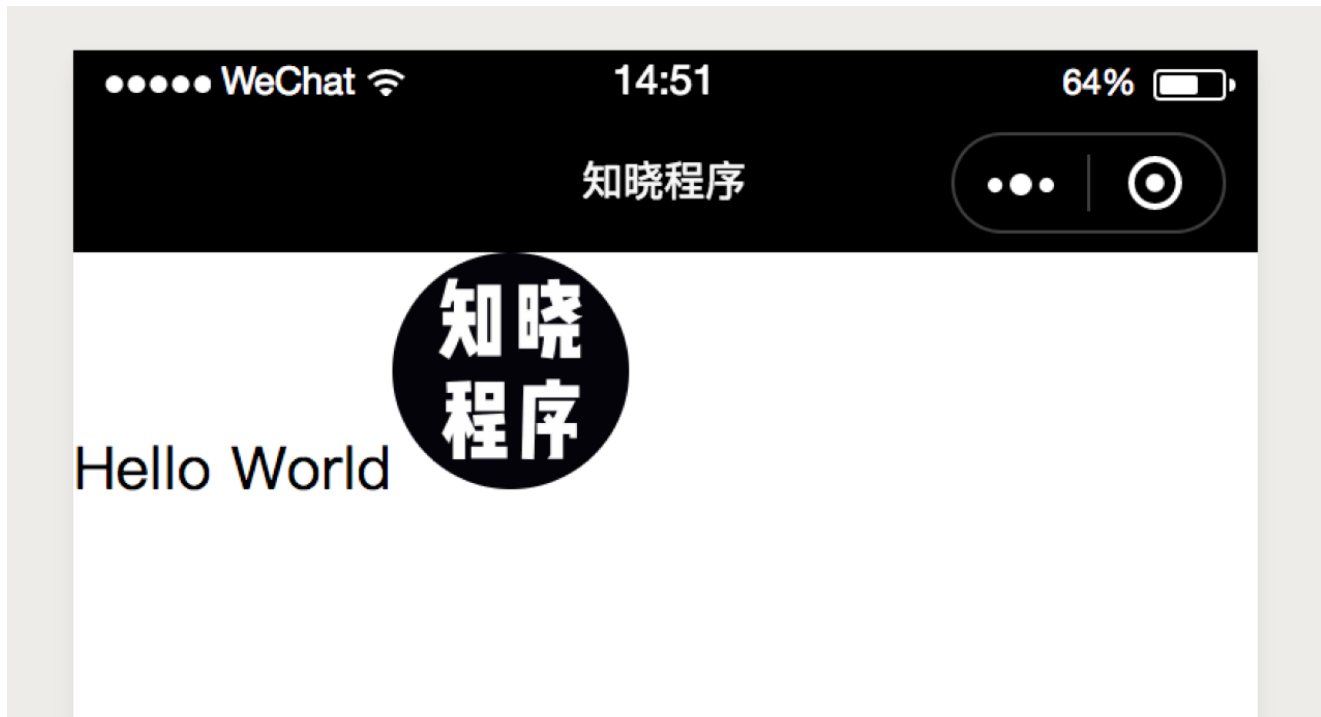
### 修改元素属性

上一章中，我们已经初步掌握 **WXSS** 的语法。这次，我们就来尝试使用 **WXSS**，来调整文字和图片大小、颜色等等属性。

找到工程中 `pages/welcome.wxss` 文件并打开，我们就可以页面元素，进行调整。先从图片开始吧：

```
image{  
  width: 150rpx;  
  border-radius: 50%;  
}
```

将这段加进去并保存，可以看到图片的大小已经被调整，图片形状也变成圆形。



调整文字字号，也可以用同样的方法进行。

```
text{  
  font-size: 64rpx;  
}
```

保存后，可以看到字号修改的结果。



## 调成元素排布

现在，我们的小程序已经有了初步形态，但它们的排布明显是不正常的。我们希望能将视觉元素以一定方法进行排列，而不是现在这样被随意放置在屏幕上。

这时候，我们就需要用到 **Flex**。什么是 **Flex** 呢？它是 **HTML** 中被广泛应用的一种视觉元素排版、排布方法，能够灵活地根据我们所需，对页面视觉元素进行排布。

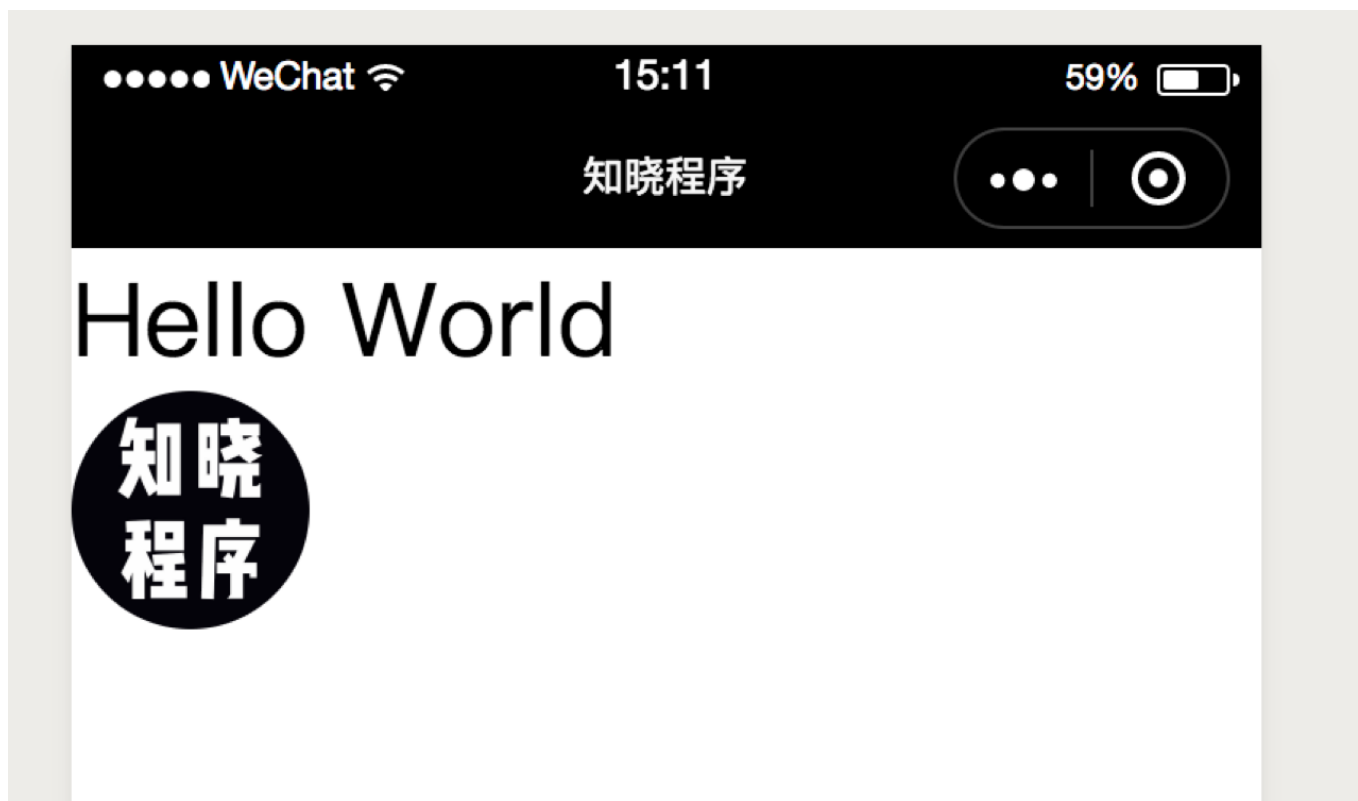
使用 **Flex** 之前，需要进行简单的声明。还记得我们在上一节中，为 `<text>` 和 `<image>` 元素「套」上的 `<view>` 吗？现在就要请它出场了。

```
view{
  display: flex;
}
```

将它添加进 `welcome.wxss` 文件中并保存，我们就可以尽情使用 **Flex** 了。

首先，我们需要让视觉元素按照「从上至下」的顺序进行排列，需要用到 `flex-direction` 属性，将 `<view>` 中的视觉元素，按照纵向进行排列。

```
view{
  display: flex;
  flex-direction: column;
}
```



接着，我们希望元素能够居中显示，这时候我们就需要用到 `align-items` 属性，它用于设定横向排版模式。将 `align-items` 值设为 `center`，就能让视觉元素居中显示。

```
view{
  display: flex;
  flex-direction: column;
  align-items: center;
}
```



另外，我们希望元素可以均匀地排布在页面上，而不是像现在这样「挤」在界面顶部。这时，我们就需要用到 `justify-content` 属性了。

```
view{
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: space-around;
}
```

这次保存，我们发现小程序界面并没有变化。先别急，我们先为 `<view>` 指定一个高度试试看。

```
view{
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: space-around;
  height: 800rpx;
}
```

保存后，我们就可以看到界面终于有了变化。

为什么为要为 `<view>` 指定高度呢？因为默认情况下，`<view>` 的高度只会与其中包裹的元素高度一致，也就是其中的文字+图片的高度。

而为 `<view>` 添加高度后，`<view>` 里的元素就会按照 `justify-content: space-around;` 属性的指示，让元素均匀地铺在整个固定高度了的 `<view>` 中。

除了这些属性之外，**Flex** 还有许多能定义的属性和值。你可以查阅相关资料，了解 **Flex** 的更多实用用法。

## 长度单位 **RPX**

也许你已经注意到，我们在 **WXSS** 中所用的长度单位并非 **PX** (**P**ixel, 像素)，而是 **RPX**。**RPX** 全称为 **responsive pixel**（直译为「动态像素」），它是微信提出的一种元素计量单位。

在 **RPX** 中，所有手机的屏幕宽度都会被固定为 **750 rpx**，实际像素以宽度 **750 rpx** 为标准进行调整。开发者不需要关心用户使用什么设备，只需使用 **rpx** 单位定义宽度，微信就能在编译时自动根据屏幕尺寸来处理元素。

微信官方提供的最佳实践是，在小程序设计过程中，使用 **iPhone 6** 的屏幕尺寸作为设计稿标准。这样，开发者在开发过程中，只需直接对尺寸除以 **2**，就能正确地配置视觉元素的尺寸。

## 为小程序加入逻辑



现在，我们的小程序已经初具雏形。接下来，我们就要为它添加逻辑，让小程序可以读取用户的昵称和头像，直接放到小程序中，变成问候语。

## 了解小程序获取用户资料的机制

在小程序中获取用户资料，方法有些特殊。我们需要先在小程序中设置一个按钮，让用户点击；然后建立一个函数，当用户点击该按钮的时候，这个函数将会接收用户资料。

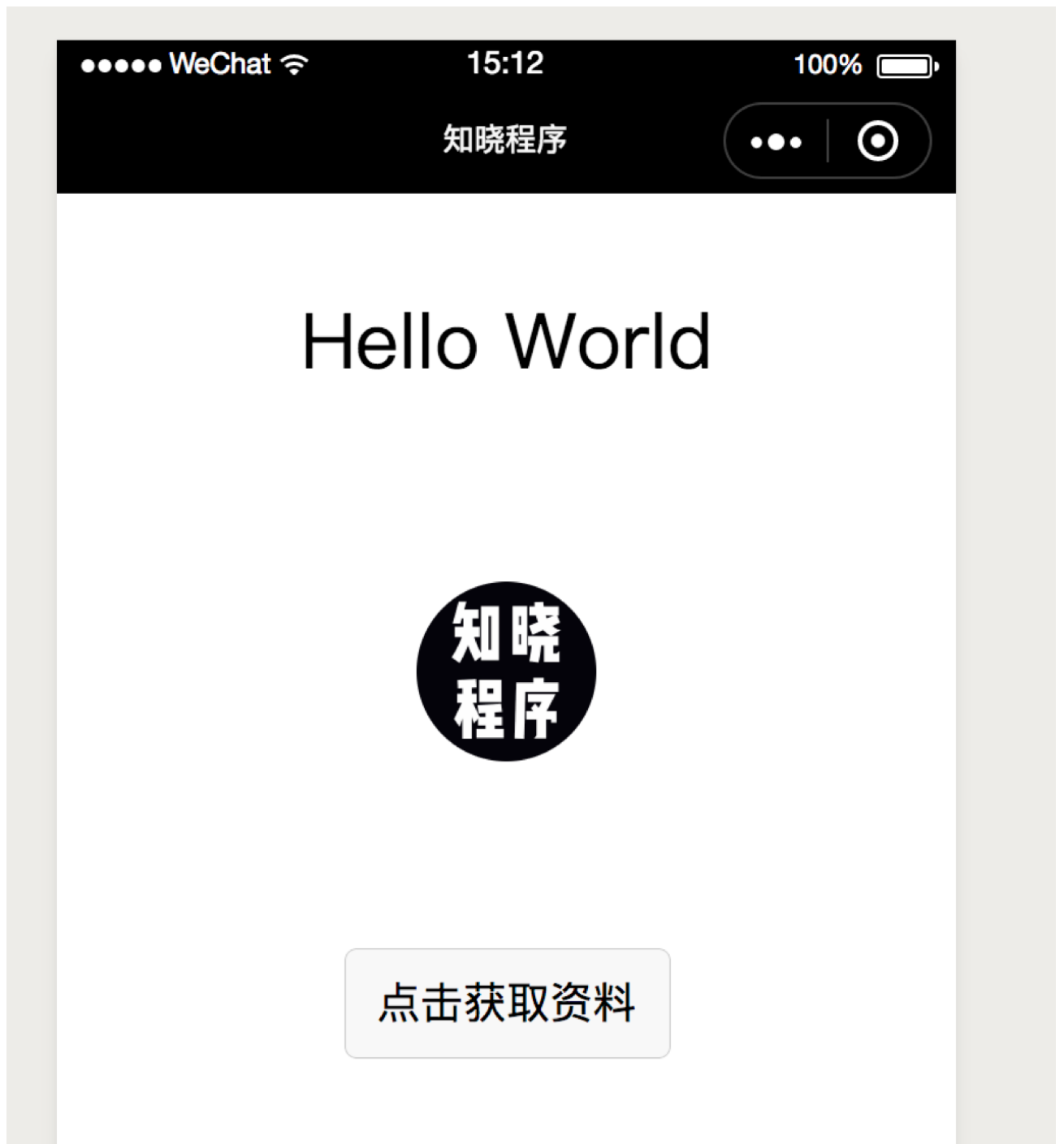
不同于其他利用接口调用用户资料的做法，微信为小程序设计这样略显奇怪的步骤，是因为微信希望当用户主动要求的时候，小程序再去请求用户资料，避免出现「刚启动小程序就要求授权」的情况。

## 新建按钮元素

回到 `index.wxml` 文件。我们在文件中加入 `<button>` 元素，然后给按钮写一个引导文字。

```
<!--pages/welcome.wxml-->
<view>
  <text>Hello World</text>
  <image src='/image/avatar.jpg' mode='widthFix'></image>
  <button>点击获取资料</button>
</view>
```

保存后，我们可以看到按钮已经在界面上了。但是当我们点击它，小程序是不会有反应的，因为微信不知道我们需要获取用户资料，放入图片和文字中。



## 从微信获取用户资料

想让按钮可以获取用户资料，我们首先要让微信知道，当用户点击按钮时，微信需要将用户资料给我们。

这时，我们需要给 `<button>` 组件添加 `open-type=getUserInfo` 属性，用以声明按钮点击的事件是获取用户资料。

```
<!--pages/welcome.wxml-->
<view>
  <text>Hello World</text>
  <image src='/image/avatar.jpg' mode='widthFix'></image>
  <button open-type='getUserInfo'>点击获取资料</button>
</view>
```

接下来，我们就要正式用上 **JS** 文件，接收微信传给我们的数据。

我们打开 `welcome.js`，可以看到微信已经自动为我们生成一些生命周期函数。我们可以直接删除其中自动生成的空白函数，只留下 `Page()` 一个函数。

```
// pages/welcome.js
Page({
})
```

接下来，我们要新建一个接收用户资料的函数，名为 `getProfile`。当这个函数接收到用户资料后，就直接输出到控制台中。

```
// // pages/welcome.js
Page({
  getProfile(res) {
    console.log(res) // console.log() 函数可以将变量、数据，写入开发者工具的控制台中。
  }
})
```

函数建好保存文件，我们还要回到 `welcome.wxml` 文件，继续修改按钮的代码，让微信将数据传入我们写好的函数中。

```
<!--pages/welcome.wxml-->
<view>
  <text>Hello World</text>
  <image src='/image/avatar.jpg' mode='widthFix'></image>
  <button open-type='getUserInfo' bindgetuserinfo='getProfile'>点击获取资料</button>
</view>
```

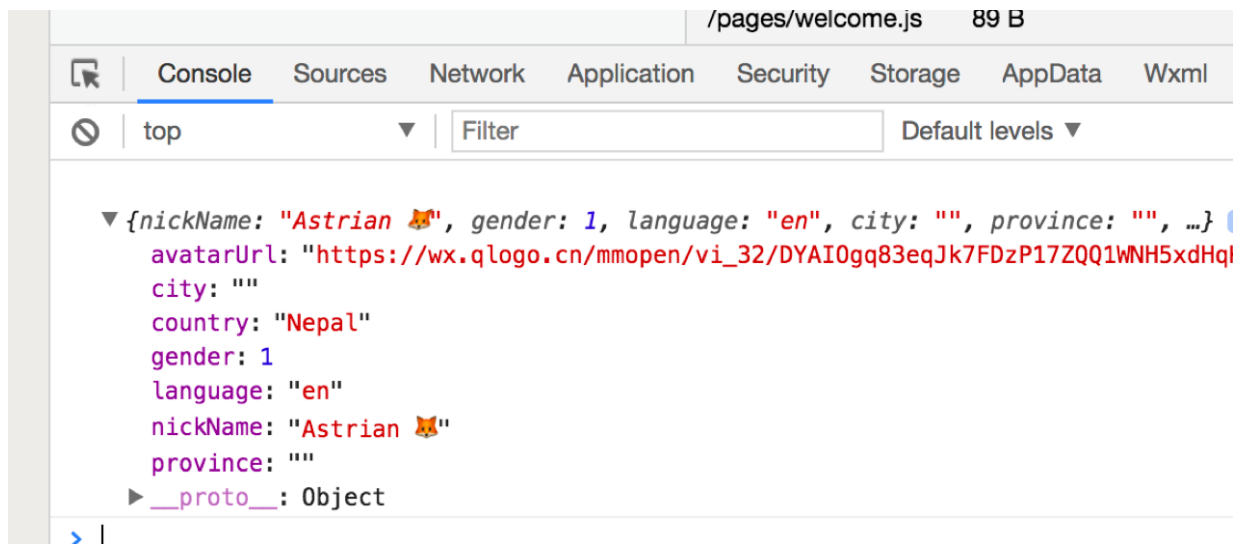
一切就绪！保存代码之后，我们在预览区尝试点击按钮。点击后，开发者工具会向我们请求授权。

点击「同意」，在调试器（开发者工具的右下角区域）的 **Console** 标签中，就可以看到我们已经成功获取用户数据。



可以看到，在这么多的数据中，微信将用户数据包裹在这个对象里的 **detail.userInfo** 中。尝试输出其中 **detail.userInfo** 的内容，就可以看到干净的用户资料了。

```
// pages/welcome.js
Page({
  getProfile(res) {
    console.log(res.detail.userInfo)
  }
})
```



## 将数据处理结果放到界面

接下来，我们就来尝试将小程序界面中原本的默认头像和欢迎语，替换为用户头像和昵称。

在小程序中，界面层（WXML）和逻辑层（JS）之间有一种特殊的数据交换方式，名为「数据绑定」。WXML 可以与 JS 中的特殊的变量进行绑定，当 JS 修改变量的时候，变化将会直接传入界面层。

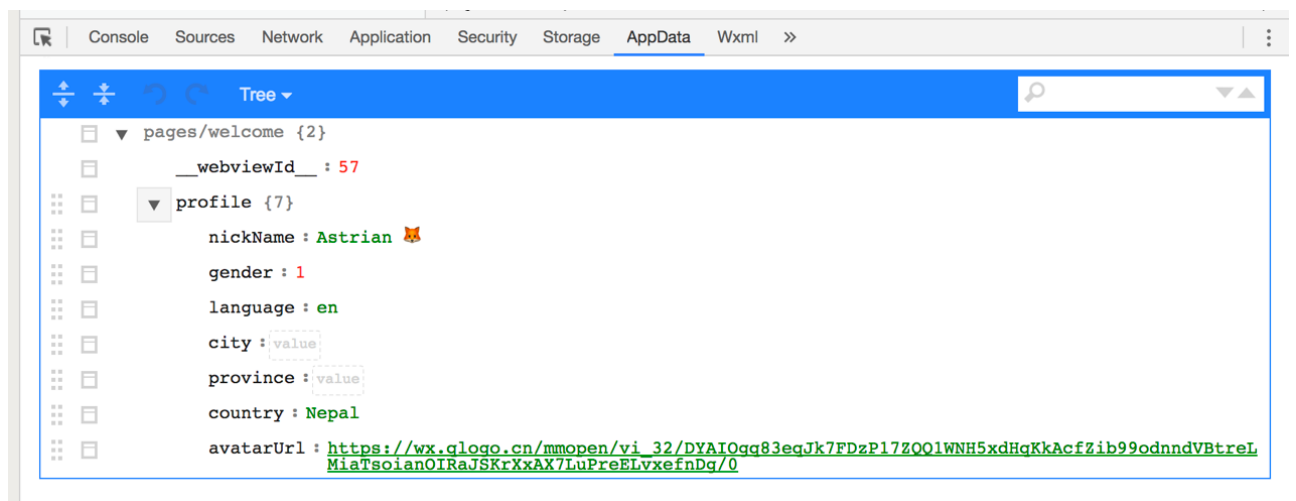
在 JS 中，想要将数据放入可在界面层展示的变量，可以利用 `Page()` 对象本身含有的 `setData()` 函数。回到 `welcome.js`，就可以用 `setData()` 将用户数据写入变量。

```
// pages/welcome.js
Page({
  getProfile(res) {
    this.setData({
      "profile": res.detail.userInfo
    })
  }
})
```

在 `wx.setData()` 函数中，我们需要传入一个 JS 对象，其中以键值对的方式，设置所需的变量。本例中，我们将用户资料，直接写入到 `profile` 变量中。

运行一下代码，再点击一次按钮，我们发现「Console」控制台不再输出获取到的用户数据（因为 `console.log()` 函数已经被删除）。那么，我们怎么知道 `setData()` 写入成功与否呢？

点击调试器的「App Data」标签，我们可以看到页面对象 `Page` 中，所有的变量。尝试重新编译小程序，点击按钮再过一会，就会发现 `profile` 变量已经写入成功了。




接下来，我们需要在 `WXML` 文件中，?绑定这些变量，以便将用户资料显示在界面上。在 `WXML` 中，我们需要用到「双花括号」的语法，对 `JS` 中的变量进行绑定操作。「双花括号」不仅可以用在小程序的 `<text>` 组件中，任何可被自由插入字符串的位置都能使用。

```
<!--pages/welcome.wxml-->
<view>
  <text>{{ profile.nickName }}</text>
  <image src='{{ profile.avatarUrl }}' mode='widthFix'></image>
  <button open-type='getUserInfo' bindgetuserinfo='getProfile'>点
  击获取资料</button>
</view>
```

重新编译小程序，再点击按钮，可以看到小程序已经成功展示用户昵称和头像。

●●●●● WeChat

17:04

100% 

知晓程序



Astrian 



点击获取资料

## 设置资料缺失时的显示

到这里，小程序已经可以很好地实现功能，但我们依然需要进行额外优化。

例如，当用户刚启动小程序时，由于小程序没有获取到我们的资料，致使整个小程序页面会是空白一片。





接下来，我们就要着手继续优化小程序，为它设置「默认数据」。让小程序获取到资料前，显示一个默认的欢迎语和图片。

如果想要在页面加载时写入初始化数据，除了利用 `setData()` 函数之外，我们还可以直接建立 `data` 对象并写入数据。这样，在加载时，`data` 对象将会自动生效。

我们在 `welcome.wxml` 中，新增 `data`，并初始化 `profile` 对象。在 `profile` 对象中，我们可以写入一些初始数据。

```
// pages/welcome.js
Page({
  getProfile(res) {
    this.setData({
      "profile": res.detail.userInfo
    })
  },
  data: {
    "profile": {
      nickName: 'Hello World',
      "avatarUrl": '/image/avatar.jpg'
    }
  }
})
```

保存代码并编译，就可以看到小程序刚启动时会自动载入初始数据。点击按钮，欢迎语和图片将会自动替换为用户资料。

Hello World



