

目 录

致谢
阅前必读
起步
代码构成
小程序能力
发布前的准备
上线
体验小程序
更新日志

致谢

当前文档《微信小程序简易教程》由 进击的皇虫 使用 书栈(BookStack.CN) 进行构建，生成于 2018-02-18。

书栈(BookStack.CN) 仅提供文档编写、整理、归类等功能，以及对文档内容的生成和导出工具。

文档内容由网友们编写和整理，书栈(BookStack.CN) 难以确认文档内容知识点是否错漏。如果您在阅读文档获取知识的时候，发现文档内容有不恰当的地方，请向我们反馈，让我们共同携手，将知识准确、高效且有效地传递给每一个人。

同时，如果您在日常生活、工作和学习中遇到有价值有营养的知识文档，欢迎分享到 书栈(BookStack.CN)，为知识的传承献上您的一份力量！

如果当前文档生成时间太久，请到 书栈(BookStack.CN) 获取最新的文档，以跟上知识更新换代的步伐。

文档地址：<http://www.bookstack.cn/books/wxadoc-quickstart>

书栈官网：<http://www.bookstack.cn>

书栈开源：<https://github.com/TruthHun>

分享，让知识传承更久远！感谢知识的创造者，感谢知识的分享者，也感谢每一位阅读到此处的读者，因为我们都将成为知识的传承者。

阅前必读

微信小程序简易教程，整理自微信官方文档。

链接地址：<https://mp.weixin.qq.com/debug/wxadoc/dev/>

起步

- 起步
 - 申请帐号
 - 安装开发工具
 - 你的第一个小程序
 - 编译预览

起步

开发小程序的第一步，你需要拥有一个小程序帐号，通过这个帐号你就可以管理你的小程序。

跟随这个教程，开始你的小程序之旅吧！

申请帐号

点击 <https://mp.weixin.qq.com/wxopen/waregister?action=step1> 根据指引填写信息和提交相应的资料，就可以拥有自己的小程序帐号。

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

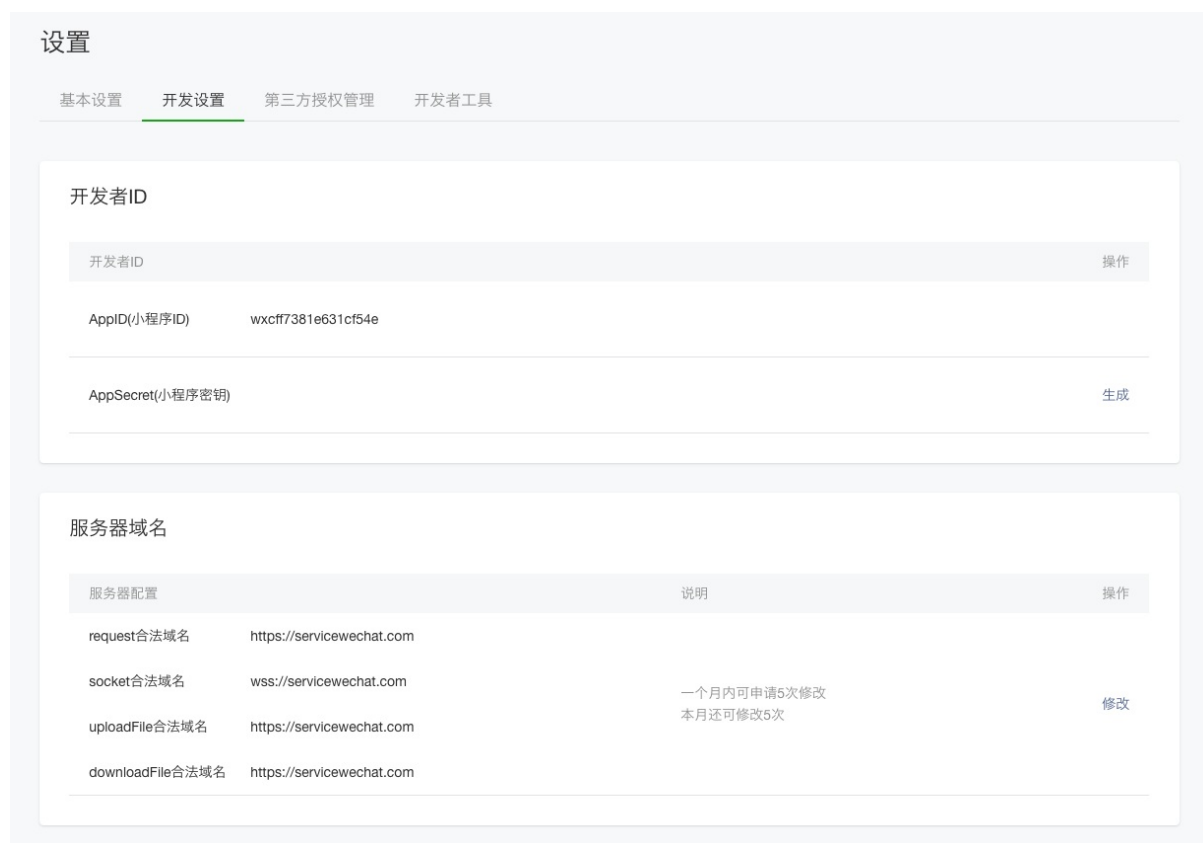
☐ 你已阅读并同意《微信公众平台服务协议》及《微信小程序平台服务条款》

注册

在这个小程序管理平台，你可以管理你的小程序的权限，查看数据报表，发布小程序等操作。

登录 <https://mp.weixin.qq.com>，我们可以在菜单“设置”-“开发设置”看到小程序的

AppID 了。



小程序的 AppID 相当于小程序平台的一个身份证，后续你会在很多地方要用到 AppID（注意这里要区别于服务号或订阅号的 AppID）。

有了小程序帐号之后，我们需要一个工具来开发小程序。

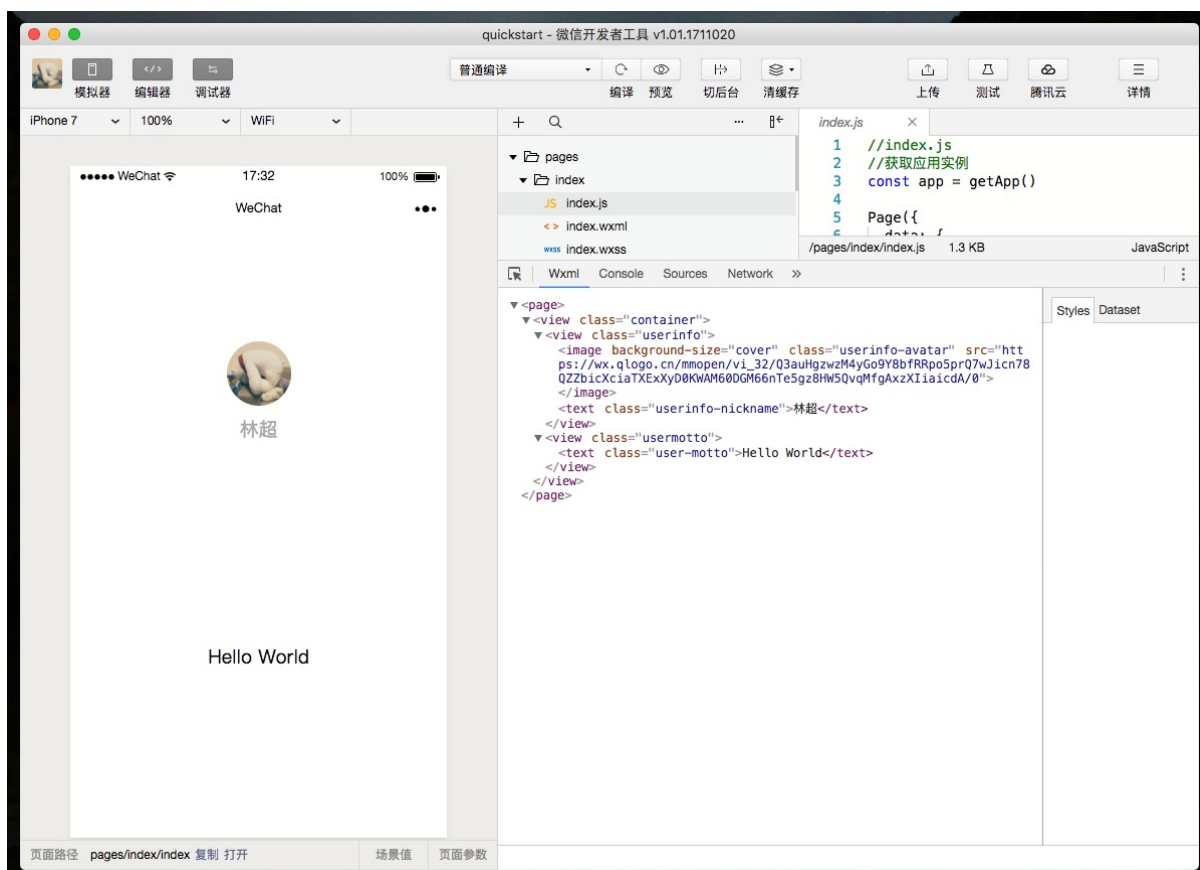
安装开发工具

前往 [开发者工具下载页面](#)，根据自己的操作系统下载对应的安装包进行安装，有关开发者工具更详细的介绍可以查看 [《开发者工具介绍》](#)。

打开小程序开发者工具，用微信扫码登录开发者工具，准备开发你的第一个小程序吧！

你的第一个小程序

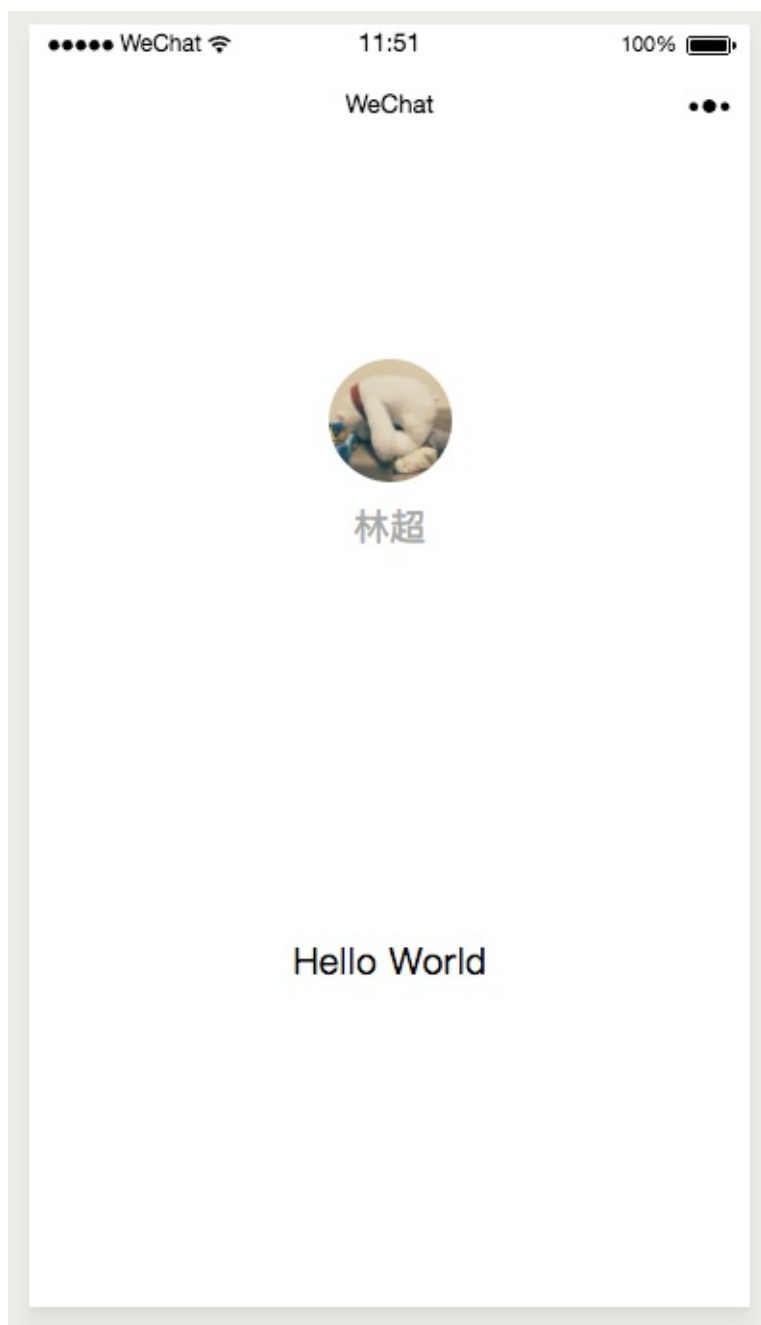
新建项目选择小程序项目，选择代码存放的硬盘路径，填入刚刚申请到的小程序的 AppID，给你的项目起一个好听的名字，最后，勾选 "创建 QuickStart 项目"（注意：你要选择一个空的目录才会有这个选项），点击确定，你就得到了你的第一个小程序了，点击顶部菜单编译就可以在 IDE 预览你的第一个小程序。



接下来我们来预览一下这个小程序的效果。

编译预览

点击工具上的编译按钮，可以在工具的左侧模拟器界面看到这个小程序的表现，也可以点击预览按钮，通过微信的扫一扫在手机上体验你的第一个小程序。



通过这个章节，你已经成功创建了你的第一个小程序，并且在微信客户端上体验到它流畅的表现。

下个章节，我们一起来看看这个小程序的代码构成。

代码构成

- 代码构成
 - JSON 配置
 - 小程序配置 app.json
 - 工具配置 project.config.json
 - 页面配置 page.json
 - WXML 模板
 - WXSS 样式
 - JS 交互逻辑

代码构成

在上一章中，我们通过开发者工具快速创建了一个 QuickStart 项目。你可以留意到这个项目里边生成了不同类型的文件：

- .json后缀的JSON配置文件
- .wxml后缀的WXML模板文件
- .wxss后缀的WXSS样式文件
- .js后缀的JS脚本逻辑文件

接下来我们分别看看这4种文件的作用。

JSON 配置

我们可以看到在项目的根目录有一个 app.json 和 project.config.json，此外在 pages/logs 目录下还有一个 logs.json，我们依次来说明一下他们的用途。

小程序配置 app.json

app.json 是对当前小程序的全局配置，包括了小程序的所有页面路径、界面表现、网络超时时间、底部 tab 等。QuickStart 项目里边的 app.json 配置内容如下：

```
1. {  
2.   "pages": [  
3.     "pages/index/index",  
4.     "pages/logs/logs"  
5.   ],  
6.   "window": {  
7.     "backgroundTextStyle": "light",  
8.     "navigationBarBackgroundColor": "#fff",  
9.     "navigationBarTitleText": "WeChat",
```



```

10.     "navigationBarTextStyle":"black"
11.   }
12. }

```

我们简单说一下这个配置各个项的含义：

- pages字段—用于描述当前小程序所有页面路径，这是为了让微信客户端知道当前你的小程序页面定义在哪个目录。
- window字段—小程序所有页面的顶部背景颜色，文字颜色定义在这里的。

其他配置项细节可以参考文档 [小程序的配置 app.json](#) 。

工具配置 project.config.json

通常大家在使用一个工具的时候，都会针对各自喜好做一些个性化配置，例如界面颜色、编译配置等等，当你换了另外一台电脑重新安装工具的时候，你还要重新配置。

考虑到这点，小程序开发者工具在每个项目的根目录都会生成一个 project.config.json，你在工具上做的任何配置都会写入到这个文件，当你重新安装工具或者换电脑工作时，你只要载入同一个项目的代码包，开发者工具就会自动帮你恢复到当时你开发项目时的个性化配置，其中会包括编辑器的颜色、代码上传时自动压缩等一系列选项。

其他配置项细节可以参考文档 [开发者工具的配置](#) 。

页面配置 page.json

这里的 page.json 其实用来表示 pages/logs 目录下的 logs.json 这类和小程序页面相关的配置。

如果你整个小程序的风格是蓝色调，那么你可以在 app.json 里边声明顶部颜色是蓝色即可。实际情况可能不是这样，可能你小程序里边的每个页面都有不一样的色调来区分不同功能模块，因此我们提供了 page.json，让开发者可以独立定义每个页面的一些属性，例如刚刚说的顶部颜色、是否允许下拉刷新等等。

其他配置项细节可以参考文档 [小程序的配置 page.json](#) 。

WXML 模板

从事过网页编程的人知道，网页编程采用的是 HTML + CSS + JS 这样的组合，其中 HTML 是用来描述当前这个页面的结构，CSS 用来描述页面的样子，JS 通常是用来处理这个页面和用户的交互。

同样道理，在小程序中也有同样的角色，其中 WXML 充当的就是类似 HTML 的角色。打开 pages/index/index.wxml，你会看到以下内容：

```

1. <view class="container">

```

```

2.   <view class="userinfo">
3.     <button wx:if="{{!hasUserInfo && canIUse}}"> 获取头像昵称 </button>
4.     <block wx:else>
5.       <image src="{{userInfo.avatarUrl}}" background-size="cover"></image>
6.       <text class="userinfo-nickname">{{userInfo.nickName}}</text>
7.     </block>
8.   </view>
9.   <view class="usermotto">
10.    <text class="user-motto">{{motto}}</text>
11.  </view>
12. </view>

```

和 HTML 非常相似，有标签、属性等等构成。但是也有很多不一样的地方，我们来一一阐述一下：

- 标签名字有点不一样

往往写HTML的时候，经常会用到的标签是div, p, span，开发者在写一个页面的时候可以根据这些基础的标签组合出不一样的组件，例如日历、弹窗等等。换个思路，既然大家都需要这些组件，为什么我们不能把这些常用的组件包装起来，大大提高我们的开发效率。

从上边的例子可以看到，小程序的WXML用的标签是view, button, text等等，这些标签就是小程序给开发者包装好的基本能力，我们还提供了地图、视频、音频等等组件能力

更多详细的组件讲述参考下个章节[小程序的能力](#)

- 多了一些wx:if这样的属性以及{{}}这样的表达式

在网页的一般开发流程中，我们通常会通过JS操作DOM(对应HTML的描述产生的树)，以引起界面的一些变化响应用户的行为。例如，用户点击某个按钮的时候，JS会记录一些状态到JS变量里边，同时通过DOMAPI操控DOM的属性或者行为，进而引起界面一些变化。当项目越来越大的时候，你的代码会充斥着非常多的界面交互逻辑和程序的各种状态变量，显然这不是一个很好的开发模式，因此就有了MVVM的开发模式(例如React, Vue)，提倡把渲染和逻辑分离。简单来说就是不要再让JS直接操控DOM，JS只需要管理状态即可，然后再通过一种模板语法来描述状态和界面结构的关系即可。

小程序的框架也是用到了这个思路，如果你需要把一个HelloWorld的字符串显示在界面上。

WXML是这么写：

```
1. <text></text>
```

JS只需要管理状态即可：

```
1. this.setData({msg:"HelloWorld"})
```

通过{{}}的语法把一个变量绑定到界面上，我们称为数据绑定。仅仅通过数据绑定还不够完整的描述状态和界面的关系，还需要if/else, for等控制能力，在小程序里边，这些控制能力都用wx:开头的属性来表达。

更详细的文档可以参考[WXML](#)

WXSS 样式

WXSS 具有 CSS 大部分的特性，小程序在 WXSS 也做了一些扩充和修改。

- 新增了尺寸单位。在写CSS样式时，开发者需要考虑到手机设备的屏幕会有不同的宽度和设备像素比，采用一些技巧来换算一些像素单位。WXSS在底层支持新的尺寸单位rpx，开发者可以免去换算的烦恼，只要交给小程序底层来换算即可，由于换算采用的浮点数运算，所以运算结果会和预期结果有一点点偏差。
- 提供了全局的样式和局部样式。和前边app.json,page.json的概念相同，你可以写一个app.wxss作为全局样式，会作用于当前小程序的所有页面，局部页面样式page.wxss仅对当前页面生效。
- 此外WXSS仅支持部分CSS选择器

更详细的文档可以参考 [WXSS](#) 。

JS 交互逻辑

一个服务仅仅只有界面展示是不够的，还需要和用户做交互：响应用户的点击、获取用户的位置等等。在小程序里边，我们就通过编写 JS 脚本文件来处理用户的操作。

```
1. <view>{{ msg }}</view>
2. <button bindtap="clickMe">点击我</button>
```

点击 button 按钮的时候，我们希望把界面上 msg 显示成 "Hello World"，于是我们在 button 上声明一个属性：bindtap，在 JS 文件里边声明了 clickMe 方法来响应这次点击操作：

```
1. Page({
2.   clickMe: function() {
3.     this.setData({ msg: "Hello World" })
4.   }
5. })
```

响应用户的操作就是这么简单，更详细的事件可以参考文档 [WXML - 事件](#) 。

此外你还可以在 JS 中调用小程序提供的丰富的 API，利用这些 API 可以很方便的调起微信提供的能力，例如获取用户信息、本地存储、微信支付等。在前边的 QuickStart 例子中，在 pages/index/index.js 就调用了 wx.getUserInfo 获取微信用户的头像和昵称，最后通过 setData 把获取到的信息显示到界面上。更多 API 可以参考文档 [小程序的API](#) 。

通过这个章节，你了解了小程序涉及到的文件类型以及对应的角色，在[下个章节]中，我们把这一章所涉及到的文件通过“小程序的框架”给“串”起来，让他们都工作起来。

小程序能力

- 小程序能力
 - 小程序的启动
 - 程序与页面
 - 组件
 - API

小程序能力

上一章中我们把小程序涉及到的文件类型阐述了一遍，我们结合 QuickStart 这个项目来讲一下这些文件是怎么配合工作的。

小程序的启动

微信客户端在打开小程序之前，会把整个小程序的代码包下载到本地。

紧接着通过 app.json 的 pages 字段就可以知道你当前小程序的所有页面路径：

```
1. {  
2.   "pages": [  
3.     "pages/index/index",  
4.     "pages/logs/logs"  
5.   ]  
6. }
```

这个配置说明在 QuickStart 项目定义了两个页面，分别位于 pages/index/index 和 pages/logs/logs 目录。而写在 pages 字段的第一个页面就是这个小程序的首页(打开小程序看到的第一个页面)。

于是微信客户端就把首页的代码装载进来，通过小程序底层的一些机制，就可以渲染出这个首页。

小程序启动之后，在 app.js 定义的 App 实例的 onLaunch 回调会被执行：

```
1. App({  
2.   onLaunch: function () {  
3.     // 小程序启动之后 触发  
4.   }  
5. })
```

整个小程序只有一个 App 实例，是全部页面共享的，更多的事件回调参考文档 [注册程序 App](#)。

接下来我们简单看看小程序的一个页面是怎么写的。

程序与页面

你可以观察到 `pages/logs/logs` 下其实是包括了4种文件的，微信客户端会先根据 `logs.json` 配置生成一个界面，顶部的颜色和文字你都可以在这个 `json` 文件里边定义好。紧接着客户端就会装载这个页面的 `WXML` 结构和 `WXSS` 样式。最后客户端会装载 `logs.js`，你可以看到 `logs.js` 的大体内容就是：

```
1. Page({
2.   data: { // 参与页面渲染的数据
3.     logs: []
4.   },
5.   onLoad: function () {
6.     // 页面渲染后 执行
7.   }
8. })
```

`Page` 是一个页面构造器，这个构造器就生成了一个页面。在生成页面的时候，小程序框架会把 `data` 数据和 `index.wxml` 一起渲染出最终的结构，于是就得到了你看到的小程序的样子。

在渲染完界面之后，页面实例就会收到一个 `onLoad` 的回调，你可以在这个回调处理你的逻辑。

有关于 `Page` 构造器更多详细的文档参考 [注册页面 Page](#)。

组件

小程序提供了丰富的基础组件给开发者，开发者可以像搭积木一样，组合各种组件拼合成自己的小程序。

就像 `HTML` 的 `div`，`p` 等标签一样，在小程序里边，你只需要在 `WXML` 写上对应的组件标签名字就可以把该组件显示在界面上，例如，你需要在界面上显示地图，你只需要这样写即可：

```
1. <map></map>
```

使用组件的时候，还可以通过属性传递值给组件，让组件可以以不同的状态去展现，例如，我们希望地图一开始的中心的经纬度是广州，那么你需要声明地图的 `longitude`(中心经度) 和 `latitude`(中心纬度) 两个属性：

```
1. <map longitude="广州经度" latitude="广州纬度"></map>
```

组件的内部行为也会通过事件的形式让开发者可以感知，例如用户点击了地图上的某个标记，你可以在 `js` 编写 `markertap` 函数来处理：

```
1. <map bindmarkertap="markertap" longitude="广州经度" latitude="广州纬度"></map>
```

当然你也可以通过 `style` 或者 `class` 来控制组件的外层样式，以便适应你的界面宽度高度等等。

更多的组件可以参考 [小程序的组件](#)。

API

为了让开发者可以很方便的调起微信提供的能力，例如获取用户信息、微信支付等等，小程序提供了很多 API 给开发者去使用。

要获取用户的地理位置时，只需要：

```
1. wx.getLocation({
2.   type: 'wgs84',
3.   success: (res) => {
4.     var latitude = res.latitude // 经度
5.     var longitude = res.longitude // 纬度
6.   }
7. })
```

调用微信扫一扫能力，只需要：

```
1. wx.scanCode({
2.   success: (res) => {
3.     console.log(res)
4.   }
5. })
```

需要注意的是：多数 API 的回调都是异步，你需要处理好代码逻辑的异步问题。

更多的 API 能力见 [小程序的API](#)。

通过这个章节你已经大概了解了小程序运行的一些基本概念，当你开发完一个小程序之后，你就需要发布你的小程序。在[下个章节]，你会知道发布前需要做什么准备。

发布前的准备

- 发布前的准备
 - 用户身份
 - 预览
 - 上传代码
 - 小程序的版本

发布前的准备

如果你只是一个人开发小程序，可以暂时先跳过这部分，如果是一个团队需要先了解一些概念。

用户身份

一个团队进行小程序的开发，那么团队成员的身份管理是很有必要的。

管理员可在小程序管理后台统一管理项目成员（包括开发者、体验者及其他成员）、设置项目成员的权限，包括：开发者/体验者权限、登录小程序管理后台、开发管理、查看小程序数据分析等。

管理入口位于：[小程序管理后台](#) - 用户身份 - 成员管理



权限	说明
开发者权限	可使用小程序开发者工具及开发版小程序进行开发
体验者权限	可使用体验版小程序
登录	可登录小程序管理后台，无需管理员确认

数据分析	使用小程序数据分析功能查看小程序数据
开发管理	小程序提交审核、发布、回退
开发设置	设置小程序服务器域名、消息推送及扫描普通链接二维码打开小程序
暂停服务设置	暂停小程序线上服务

预览

使用开发工具可以预览小程序，帮助开发者检查小程序在移动客户端上的真实表现。

点击开发者工具顶部操作栏的预览按钮，开发工具会自动打包当前项目，并上传小程序代码至微信的服务器，成功之后会在界面上显示一个二维码。使用当前小程序开发者的微信扫码即可看到小程序在手机客户端上的真实表现。

上传代码

同预览不同，上传代码是用于提交体验或者审核使用的。

点击开发者工具顶部操作栏的上传按钮，填写版本号以及项目备注，需要注意的是，这里版本号以及项目备注是为了方便管理员检查版本使用的，开发者可以根据自己的实际要求来填写这两个字段。

上传成功之后，登录[小程序管理后台](#) - 开发管理 - 开发版本 就可以找到刚提交上传的版本了。

可以将这个版本设置 体验版 或者是 提交审核

小程序的版本

版本	说明
开发版本	使用开发者工具，可将代码上传到开发版本中。 开发版本只保留每人最新的一份上传的代码。点击提交审核，可将代码提交审核。开发版本可删除，不影响线上版本和审核中版本的代码。
审核中版本	只能有一份代码处于审核中。有审核结果后可以发布到线上，也可直接重新提交审核，覆盖原审核版本。
线上版本	线上所有用户使用的代码版本，该版本代码在新版本代码发布后被覆盖更新。

可以使用小程序开发者助手方便快捷的预览和体验线上版本，体验版本以及开发版本。



在[下个章节]，我们来看看如何发布一个小程序，让你的成果被所有的微信用户所使用到。

上线

- [上线](#)
 - [提交审核](#)
 - [发布](#)
 - [运营数据](#)

上线

开发和测试完成之后，就需要将小程序发布给用户来使用了。

提交审核

为了保证小程序的质量，以及符合相关的规范，小程序的发布是需要经过审核的

在开发者工具中上传了小程序代码之后，登录 [小程序管理后台](#) - 开发管理 - 开发版本 找到提交上传的版本。

在开发版本的列表中，点击 提交审核 按照页面提示，填写相关的信息，即可以将小程序提交审核。

需要注意的是，请开发者严格测试了版本之后，再提交审核， 过多的审核不通过，可能会影响后续的时间。

发布

审核通过之后，管理员的微信中会收到小程序通过审核的通知，此时登录 [小程序管理后台](#) - 开发管理 - 审核版本。中可以看到通过审核的版本。

请点击发布，既可发布小程序。

运营数据

有两种方式可以方便的看到小程序的 [运营数据](#)

方法一：

登录 [小程序管理后台](#) - 数据分析

点击相应的 tab 可以看到相关的数据。

方法二：

使用小程序数据助手，在微信中方便的查看运营数据



体验小程序

体验

下载微信客户端版本号：6.3.27 及以上 [下载源码](#) 版本20170111





更新日志

- [更新日志](#)
 - [2018.02.08 开发者工具更新](#)
 - [历史更新日志](#)

更新日志

2018.02.08 开发者工具更新

- **A** 新增iPhoneX尺寸
- **U** 更新自动补全更新
- **F** 修复多开时登录态丢失问题[详情](#)
- **F** 修复Windows某些版本下工具可能打不开的问题[详情](#)
- **F** 修复硬盘空间满的时候，如果保存出错文件会被清空的问题[详情](#)
- **F** 修复 `wx.getSystemInfo` 返回缺少 `batteryLevel` 的问题
- **F** 修复部分API自动补全格式错误[详情](#)
- **F** 修复 `wx.navigateBackMiniProgram` 返回信息错误问题[详情](#)
- **F** 修复焦点不在编辑器时，按下跳转文件快捷键没有自动让它获取到焦点的问题
- **F** 修复在 `app.json` 编辑 `pages` 自动创建文件时，路径逾越了项目路径也能成功的问题
- **F** 修复 `app.json` 中 `enablePullDownRefresh` 没有严格校验类型的问题[详情](#)

历史更新日志

[历史更新日志](#)