





阿里云开发者“藏经阁”  
海量免费电子书下载

## 顾问

侯发欣（铁欣）

## 策划编辑

徐雯昀（畹汀）

## 主编

徐雯昀（畹汀）、纪宇鹏（风隐）、胡可茹、郭安然、袁佳茗、  
蔡伟伟、郑智宇、郭兴越

## 设计

回祎宁

欢迎扫码入群，交流阅读心得



# 目录

## Contents

## 小程序开发不求人

|                                   |            |
|-----------------------------------|------------|
| <b>第一章 支付宝小程序介绍及开发准备 .....</b>    | <b>4</b>   |
| 介绍 .....                          | 5          |
| 小程序开发准备 .....                     | 10         |
| <b>第二章 支付宝小程序开发者工具介绍 .....</b>    | <b>12</b>  |
| <b>第三章 支付宝小程序开发基础知识 .....</b>     | <b>21</b>  |
| 支付宝小程序代码结构及组成 .....               | 22         |
| 支付宝小程序 API .....                  | 60         |
| 支付宝小程序组件 .....                    | 81         |
| 支付宝小程序开放能力 .....                  | 103        |
| <b>第四章 支付宝小程序快速示例 .....</b>       | <b>112</b> |
| <b>第五章 小程序协同工作及版本管理 .....</b>     | <b>119</b> |
| 小程序成员管理 .....                     | 120        |
| 小程序版本管理 .....                     | 125        |
| <b>第六章 小程序基础库更新迭代 .....</b>       | <b>128</b> |
| 基础库更新迭代 .....                     | 129        |
| <b>第七章 IoT 小程序开发及刷脸支付实现 .....</b> | <b>132</b> |
| IoT 小程序开发及刷脸支付实现 .....            | 133        |
| <b>第八章 小程序模板及插件开发应用 .....</b>     | <b>135</b> |
| 小程序模板介绍 .....                     | 136        |
| 小程序插件介绍 .....                     | 138        |

# Chapter 1

## 第一章 支付宝小程序介绍及开发准备



# 介绍

## 支付宝小程序为何如此重要？

小程序即用即走、无需下载的特性为大量用户提供了更便捷的使用方式。如今的支付宝 App 已是移动互联网生态的超级 App 之一，小程序这种轻量化的解决方案，可以让更多的外部开发者通过支付宝来服务更多客户，同时也可以让支付宝平台拓展出更多的开放服务能力，以满足用户多样化的需求。

早在 2017 年，当支付宝小程序还只是一堆代码时，支付宝就曾明确谈过它的定位：“用互联网的技术、产品帮助商家更好地服务用户”。

2019 年 9 月 17 日，在“支付宝开放日·小程序年度峰会”上，阿里巴巴合伙人、支付宝事业群总裁倪行军（苗人凤）透露，支付宝平台已累计上线 100 多万个小程序，月活用户突破 5 亿。更重要的是，和微信小程序集中于游戏、直播领域的情况不同，支付宝百万级的小程序中，有 70%集中于线下的商业生活服务领域，这恰好是离钱最近的地方。

## 支付宝小程序是怎样做到的？

### 基于商家需求和平台特性，打造运营玩法

除了 C 端用户的支付和生活服务需求之外，支付宝多年来还积累了几千万的 B 端商户，希望更好地服务这些客户。而小程序跟大平台的完美结合，无疑大大满足了商家对流量获取、精准营销、提高用户留存率等核心需求。

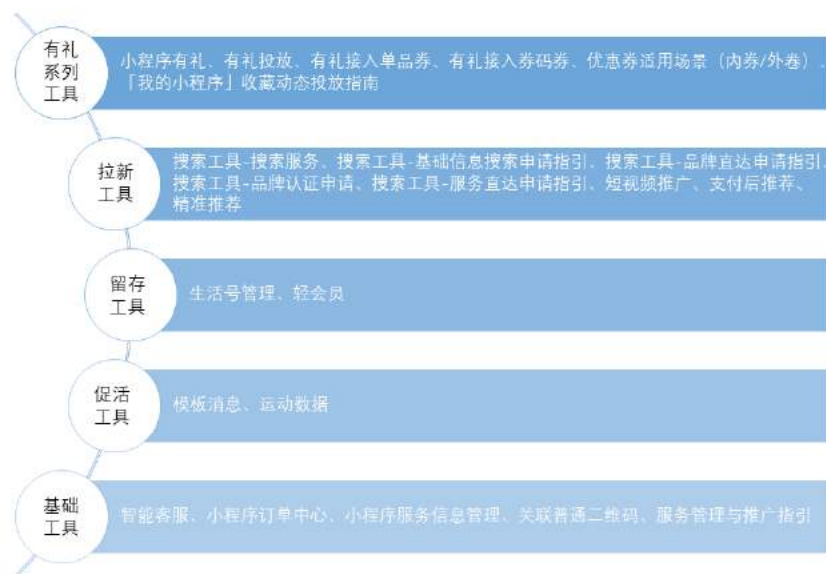
支付宝小程序在运营玩法中提供包括了支付宝中心化频道触达用户机会的官方流量奖励，和支持在支付宝 APP 端开放链接用户的入口的自运营工具支持。

### 中心化频道开放

支付宝鼓励生态商家在端内持续运营小程序，推出了长期的流量奖励政策。一起分享支付宝中心化频道触达用户机会，支付宝首页、会员频道、惠支付、花呗频道、信用频道、社区生活等更多的频道逐步开放。

### 自运营入口开放

在支付宝 App 端开放链接用户的入口，更便捷地让合作伙伴结合自身运营诉求进行自运营，支付成功页、生活号、朋友 TAB、小程序收藏、搜索栏、账单页、二维码扫一扫、卡包等多个支付宝端内入口和工具，帮助商家打造拉新留存促活的运营闭环。



以“轻会员”为例。说是会员，实际上是一种特殊的会员营销工具。通过结合芝麻信用与花呗特性，“轻会员”可以让消费者无需预付任何费用即可成为会员，先享受优惠权益，到期后再结算会员费，最核心的特点是可以先享后付。之所以“轻”，是因为配置即可上线，而且没有底层的后台，更多是会员营销方式的改变。例如知名茶饮品牌“奈雪的茶”，在“奈雪点单”小程序上，办理“轻会员”的用户比未办理用户下单量高出 42%，客单价提升了 68%，基本上算是一个超级用户的雏形。相对而言，这种会员模式对 B 端和 C 端都能形成一种对等的保护：帮 B 促营收，刺激 C 消费。对于商家来说，可以快速刺激回头客，拉复购。

## “一云多端”背后的阿里开放生态

阿里巴巴正在做的事情是：全面拥抱小程序，为小程序提供全面的技术、业务、生态的支持，能够帮助我们的企业在未来的云生态里面走的更远。

小程序云是面向小程序场景提供的一站式云服务，帮助开发者实现一云多端的业务战略，提供了有服务器和无服务器两种模式。跨端开发工具链为开发者提供了一次开发全网小程序运行的能力，并在一朵云内实现统一的资源管理、统一的数据运营和统一的业务设计。

小程序多端发布则与商家一起分享阿里集团多端触达用户的机会，手机淘宝、高德、UC 浏览器等阿里集团更多亿级用户 APP 逐步开放。开发者可使用同一套代码，发布到阿里开放生态的各个端，降低开发成本。一次开发、多端投放，全方位享受阿里经济体红利的基础正在成型。通过阿里小程序矩阵，商家也可以打造品牌效应、提高用户粘性，获取更大的商业价值。

同时，基于支付宝开放平台的战略，小程序可以将开发者背后大量的第三方企业的开发能力聚合到平台，以此加固阿里的生态体，最终成为其构建多元化盈利渠道的重要工具和竞争手段。

未来，阿里系的几大超级 App，包括淘宝、钉钉、高德、饿了么等，底层能力都将打通，形成无缝对接。

这意味着，阿里系的多元化能力，可以随意组合了。



比如，小程序不仅可以拥有支付宝的花呗分期能力，还可以拥有菜鸟的物流能力、饿了么的配送能力、高德的 LBS 能力等等。这些能力都以开放能力功能包的形式汇聚到一个开放能力市场里。开发者在开发应用的时候，可以方便地获取功能包，接入对应的 API，进行“拼积木式组装”。这相当于集众家之长，组成一个神通广大的“混血儿”。



举个例子，用户想吃火锅，便在高德搜索附近的海底捞，然后开车去，导航结束时，高德 App 会自动弹出海底捞支付宝小程序，用户可以直接进行点餐、领优惠券；亦或者，用户想去商场逛街，导航结束时，高德 App 弹出相应商场的支付宝



小程序，小程序里面有停车服务、有商场里面一些餐厅、服装店的优惠券，用户可以快速领取。

目前，以上两种场景已经实现。虽然阿里系小程序还处于萌芽早期，落地层面案例还不多，不过随着各大超级 App 底层能力的互通，势必会实现多场景的打通和串联。

## 当谈起支付宝小程序，我们是在畅想怎样的未来？

新的时代需要新的工具设施。小程序的诞生与兴盛，是当下阶段，人们加速拥抱移动互联网时代、拥抱碎片化消费时代的集中反映。

未来小程序这个载体会变，但数字时代数字化经营的大基础不会变。

2018 年，时任阿里巴巴集团首席执行官的张勇，提出阿里的新目标是要做商业操作系统——“在阿里巴巴经济体中，包括购物、娱乐、本地生活等多元化的商业场景及其形成的数据资产，与阿里巴巴正在高速推进的云计算一经结合，共同形成了独特的阿里巴巴商业操作系统。在这个操作系统当中，各个商业部门既产生数据，又运用数据，形成一个庞大而丰富的有机循环。”

支付宝小程序的崛起，是对阿里商业操作系统的正式落地。加之，小程序有三个特色——钱、服务、信任；两个聚焦——商业和生活服务；一个坚持——坚持安全可靠。这些正是阿里商业操作系统所需要的。



在阿里商业操作系统的构想中，阿里生态将为企业输出一整套的数字化能力，而非提供单一工具。阿里经济体内的品牌、商品、销售、营销、渠道、制造、服务、金

融、物流供应链、组织、信息管理系统等企业运营中的 11 大商业要素，都会融合性地助力零售业的数字化转型，改造商业结构、创造出新的需求和市场；而集成这 11 种商业要素的最佳载体，就是小程序。

小程序是阿里打造数字经济体的重要组成部分，将坚持开放路线，不管是流量、能力还是模式，跑通一个，开放一个，成熟一个，开放一个。阿里巴巴的使命是让天下没有难做的生意，支付宝小程序的初心亦是如此。

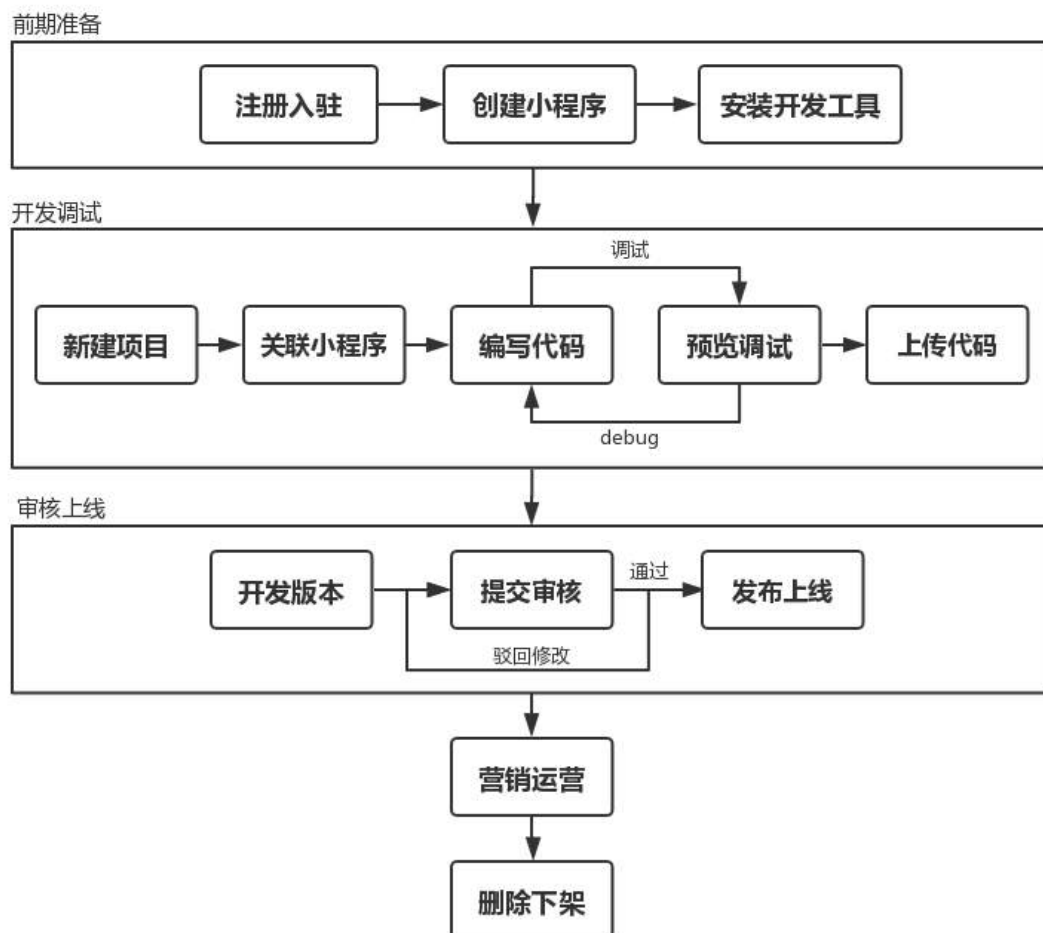
在阿里巴巴 20 周年的演讲中，新任阿里巴巴集团董事局主席张勇说：“我们今天处在最好的时代，因为这个时代，才有 20 年阿里巴巴的发展。我们要感恩，我们希望不断为社会创造价值，为社会承担更多的责任。”

“如果，因为我们的努力，社会能进步；因为努力，商家能更好；我们将由衷高兴。我们希望客户、合作伙伴过得比我们好。”

# 小程序开发准备

作为一名新入门的开发者，想要从开发小程序到开展业务，我们需要经历哪些阶段呢？

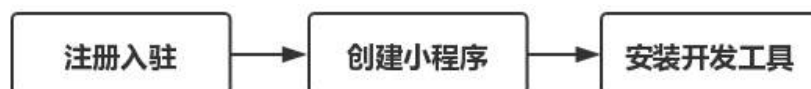
通过下图所示，您可以大致了解小程序的开发流程阶段。



## 开发准备

本章将着重为大家介绍小程序的开发前期准备工作。

在开发小程序之前，您需要完成下列准备工作：



### 一、注册入驻

您需要拥有支付宝账户，然后以开发者身份入驻 [支付宝开放平台](#)，才能创建小程序。

## 二、创建小程序

在实际开发之前，您需要在后台创建空白小程序。此步骤仅起到命名作用，小程序的实际内容还需后续开发。

**创建步骤：**

1. 登录 [小程序开发中心](#)，可以看到 **我的小程序** 页面。
2. 在 **我的小程序** 页面点击 **开始创建**。
3. 填写 **小程序名称**，点击 **创建**。

## 三、安装开发工具

下载并安装 [小程序开发者工具](#)（简称 IDE）的最新版本。

请根据操作系统选择对应的开发工具：Windows 64 位或 MacOS。其它操作系统下暂时未提供开发工具。

# Chapter 2

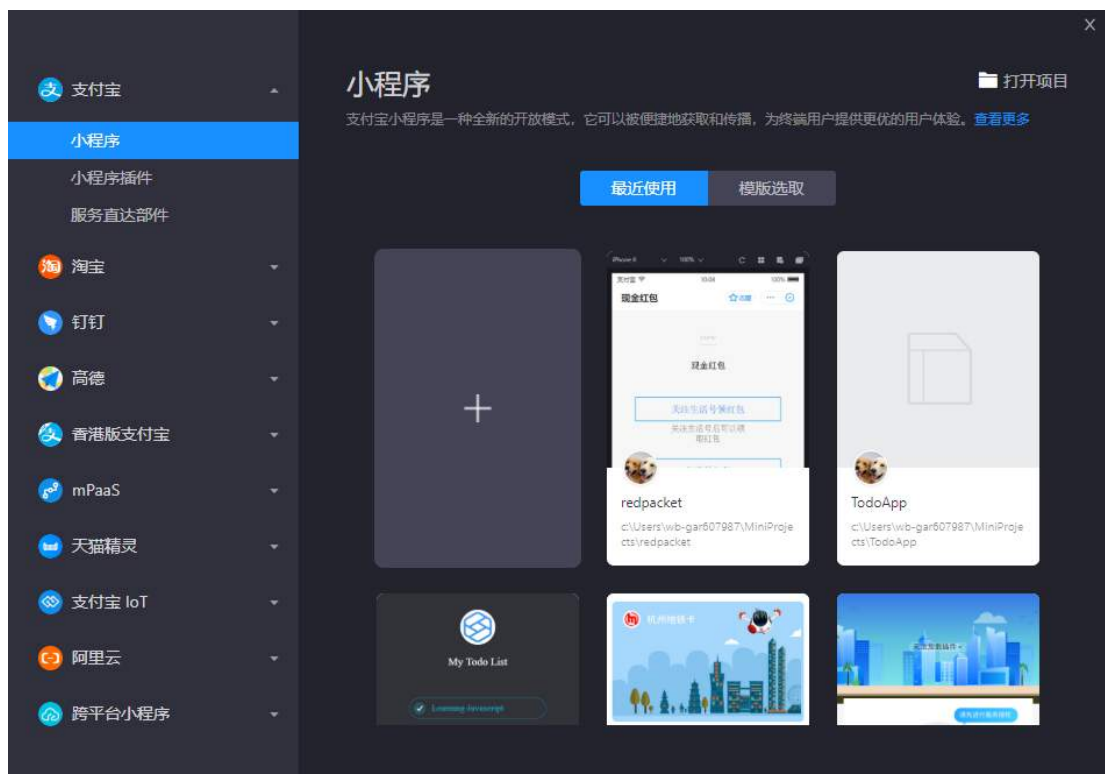
## 第二章 支付宝小程序开发者工具介绍



# 界面介绍

## 启动界面

在小程序开发者工具（简称 IDE）的启动界面，开发者可以新建项目、打开现有项目、删除最近项目记录。



## 新建项目

新建项目类型分为两类：**示例模板** 与 **空白脚手架**。

### 示例模板项目

开放平台提供入门、UI、开放能力三类模板，内含大量示例代码（仍在持续更新中），为开发者演示如何实现小程序各项能力（参见 [快速示例](#)）。新建步骤如下：

1. 点击 **模板选取** 标签。
2. 点击需要创建的模板，然后点击 **下一步**；或者直接双击所需模板卡片。
3. 在 **新建项目** 页面：
  - a. 设置 **项目名称** 与 **项目路径**；
  - b. 点击 **完成**，进入主界面。

## 空白脚手架项目

空白脚手架仅包含最基础的文件结构。新建步骤如下：

1. 在左侧边栏选择 **项目类型**（例如支付宝小程序、淘宝商家应用等类型）。
2. 在 **最近使用** 列表中，点击 **+** 卡片。
3. 在 **新建项目** 页面：
  - a. 设置 **项目名称** 与 **项目路径**；
  - b. 选配 **后端服务**：不启用云服务、小程序 Serverless 或 小程序云应用；
  - c. 点击 **完成**，进入主界面。

## 打开项目

本地现有项目也有两种打开方式：**选择最近项目** 与 **选择项目文件夹**。

### 选择最近项目

在启动界面 **最近使用** 列表中，点击需要打开的项目，然后点击 **打开**；或者直接双击所需项目卡片。

### 选择项目文件夹

1. 点击右上角的 **打开项目**，弹出文件窗口。
2. 导航至项目文件夹，点击 **选择文件夹**。
3. 在 **打开本地项目** 页面：
  - a. 确认或修改 **项目名称** 与 **项目类型**；
  - b. 点击 **打开**，进入主界面。

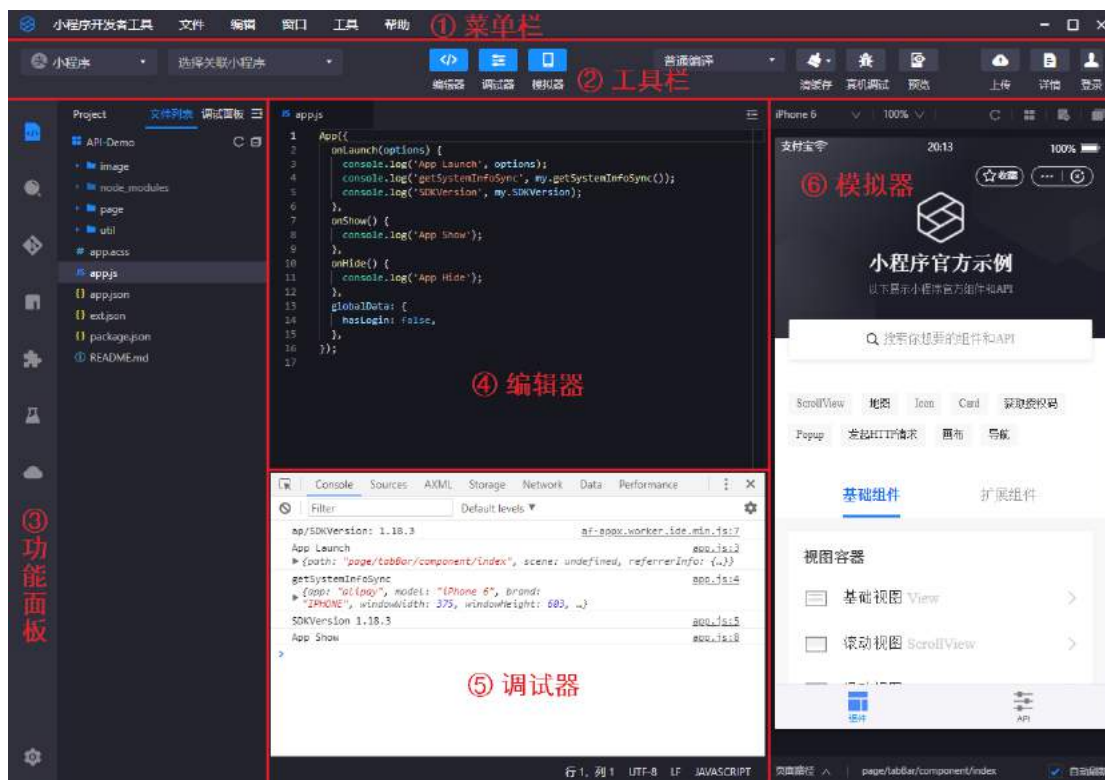
## 删除最近项目记录

如需维持启动界面整洁，首先右键点击希望清理的项目卡片，然后点击 **删除**。

**注意：**此项操作仅会清理启动界面记录，不会实际删除硬盘中的项目文件。

## 主界面

小程序开发者工具的主界面由 **菜单栏**、**工具栏**、**功能面板**、**编辑器**、**调试器**、**模拟器** 组成。



主界面各区域简要说明：

1. **菜单栏**：文件、编辑、窗口、工具、帮助等基础功能。
2. **工具栏**：更改小程序类型与关联、显示/隐藏界面区域、真机调试与预览等功能。
3. **功能面板**：切换文件树、搜索、git 管理、npm 管理、插件市场、实验室、云服务等边栏。
4. **编辑器**：输入与修改代码的区域。
5. **调试器**：用于模拟器调试、真机调试、性能调试。
6. **模拟器**：模拟运行小程序，便于快速预览、初步调试。

说明：使用工具栏中部三个按钮，可以显示或隐藏界面主要区域，其中 **编辑器** 与 **调试器** 无法同时隐藏。

## 工具栏

小程序开发者工具（简称 IDE）的工具栏位于主界面顶部，包含 IDE 中最常用的主要功能。本文档从左至右介绍各项功能。

## 小程序



选择小程序的运行环境类型（默认为支付宝小程序），此外还支持淘宝、钉钉、高德等运行环境（参见 [多端支持](#)）。

## 关联小程序

每个开发者账号可以拥有多个小程序的开发权限，因此需要关联具体小程序，决定小程序代码的上传位置。

在此项下拉列表中，点击 **+ 创建小程序**，进入小程序管理页面，可以创建小程序供后续关联。

**说明：**在创建小程序命名时，注意遵循名称规范。


如需关联其他账号中的小程序，但开发者账号还没有开发权限的话：

1. 使用主账号登录小程序开发中心，点击所需小程序，跳转至开发管理页面。
2. 在左侧导航栏选择 **成员管理**，点击 **添加**，按照提示添加开发成员。
3. 使用开发者账号在支付宝客户端 **朋友 > 服务提醒** 中接受邀请。
4. 使用开发者账号登录 IDE，关联小程序。

## 关联云服务

**说明：**在最初新建项目环节，如果后端选用云应用，此项功能才会显示。

如果未启用云服务或者选用 Serverless，此项功能则会隐藏。

每个小程序可以创建多个云应用，而云应用又包含测试与正式环境。因此需要关联具体的云应用环境，决定云应用代码的上传位置。在关联云应用环境之后，此项下拉列表右侧会出现  按钮，用于上传代码与管理云应用。

在此项下拉列表中，点击 **+ 创建新服务**，进入云服务管理页面，可以创建云应用后续关联。



## 显示/隐藏界面区域

点击 **编辑器**、**调试器**、**模拟器**，可以分别显示与隐藏这三个区域。

**说明：**功能面板/编辑器 会同时显示/隐藏，而 编辑器/调试器 无法同时隐藏。



## 编译模式

在普通编译模式下，小程序初始的启动页面是 app.json 文件 pages 列表的首个页面，且不带任何参数。

如需快速调试其他页面 或 设置参数：

1. 在此项下拉列表中点击 **+** 添加编译模式。
2. 在弹窗中填写 **模式名称**、**页面参数**、**全局参数**，选择 **启动页面**，点击 **确定**。
3. 随后模拟器将会改用新的启动页面，同时自动传入设置好的参数。

**说明：**弹窗中如果勾选 **下次编译时模拟更新**，可以模拟小程序更新效果，详情请参见 [UpdateManager](#)。

## 清除缓存

清除 **数据缓存**、**授权数据**、**文件缓存**、**构建缓存**、**网络缓存**。

## 真机调试

通过 IDE 远程至真机，设置断点、查看运行信息。详情请参见 [真机调试](#)。

**说明：**如不希望频繁扫码，可以使用 **自动预览** 选项。

## 预览

使用真机预览小程序，初步查看 API 真机调用效果。

**说明：**如不希望频繁扫码，可以使用 **自动预览** 选项。

## 上传

上传项目代码至关联小程序的后台空间。在小程序完成开发之后，项目代码需要打包上传至开放平台，然后才能提交审核与发布上线。目前只能上传图片不支持上传视频。

### 前提条件

在上传代码之前，本地项目需要关联至后台小程序，用于确定上传位置。如果尚未关联，点击工具栏右侧的 **上传** 菜单时，会有弹窗提示进行关联。IDE 中选择的小程序与后台查看版本详情的小程序需要保持一致。

### 上传方法

1. 在工具栏右侧的 **上传** 菜单栏中，点击 **上传** 按钮。
2. 上传成功之后，后台会生成新的开发版本条目。查看方法：登录 [小程序开发中心](#)，点击所需小程序，版本条目会在开发管理页面的 **版本详情** 区域显示。

上传之前的选项：

- **上传版本**：每次上传时版本默认递增 0.0.1（本次版本必须大于线上版本），从而确保后台每份代码版本唯一。
- **创建预审核任务**：免费调用一台真机进行测试（每天 5 次限额），详情请参见 [预审核](#)。

## 配置域名白名单

在上传之前阶段（模拟器预览/调试、真机预览/调试），小程序默认不会限制域名访问；但在上传之后阶段（体验版本、审核版本、灰度版本、线上版本），小程序只能访问白名单域名。若未成功配置白名单，可能会导致小程序页面白屏。详细操作可参见 [配置 H5 服务器域名白名单](#)。

配置步骤：

1. 登录 [小程序开发中心](#)，选择所需小程序。
2. 在左侧导航栏点击 **设置**，进入设置页面。
3. 点击 **开发设置** 标签，在 **服务器域名白名单** 区域点击 **添加**。
4. 填写所需域名，点击 **确定**。

**注意**：添加的域名必须支持 HTTPS 协议，而且已经完成备案。

## 详情

- 查看关联应用名称、项目本地目录、线上版本。
- 选择是否启用自定义组件 component2 编译、axml 严格语法检查。
- 查看 my.request、web-view 域名白名单信息。  
设置是否忽略这两项域名检查。

**注意**：小程序 **体验版** 与 **提审之后版本** 无法继续忽略检查，届时请务必设置域名白名单！

域名白名单设置入口有两处：

- 在详情页面，点击 **域名信息** 右侧蓝色按钮，进入设置页面，点击 **开发设置**。
- 登录 **我的小程序**，选择所需小程序；从左侧边栏进入 **设置 > 开发设置**。

## 登录/用户头像

- 在开发者未登录时，显示 **登录** 按钮。
- 在开发者登录后，显示开发者头像。可以查看消息通知，或者退出登录。



## 模拟器

小程序项目通过编译之后，自动在模拟器中运行，无需真机即可快速预览。基础互动方式是通过鼠标点击、拖拽来模拟手指触摸、拖动操作。

在默认设置下，每次保存已变更代码时都会触发模拟器自动刷新，实现准实时预览效果。关闭自动刷新的方法：取消勾选模拟器右下角 **自动更新**。

## 模拟器布局

模拟器顶部各项功能（从左至右）：

- **设备尺寸**：选择预设的 iOS 或安卓设备尺寸，或者新建自定义的设备尺寸，用于测试适配性。
- **缩放比例**：控制小程序的显示缩放比例。
- **刷新**：重新编译代码并刷新模拟器。
- **小工具**：打开/关闭模拟器的小工具菜单。
- **模拟器日志**：在编辑器打开模拟器日志窗口。
- **独立窗口**：改用独立窗口方式显示模拟器。

在独立窗口模式下，界面新增 **窗口置顶** 按钮；**独立窗口** 变为 **合并窗口** 按钮。

模拟器底部三项功能：

- **页面路径**：显示当前页面路径。点击路径，可直接打开当前页面 js 文件。
- **页面参数**：显示当前页面收到的参数。点击参数，可快速复制至剪贴板。
- **自动刷新**：如需避免频繁刷新，可取消勾选此功能。

说明：点击 ^ 可以切换显示 页面路径 或 页面参数。

## 模拟器小工具

点击图中蓝色小工具图标，显示/隐藏更多模拟功能：



首页：模拟按下 Home 键，小程序退回后台。用于测试生命周期函数 `onShow` 与 `onHide`，参见 小程序运行机制。

定位：输入设置 **经纬度** 模拟信息，用于测试 `my.getLocation` 获取用户位置端口（经纬度模拟信息需要输入浮点数，精确至 1 位小数以上）。

扫码：输入设置 扫码返回数据 模拟信息，用于测试 `my.scan` 扫一扫端口。

摇一摇：模拟摇一摇动作，用于测试 `my.watchShake` 摇一摇端口。

授权：设置 **用户信息、地理位置、相册、相机、麦克风** 模拟权限，可用于测试 `my.getLocation`、`my.chooseImage` 等端口的权限请求步骤（在已获得用户信息授权时，还提供删除用户信息授权的选项）。

裁剪：模拟用户截屏情况，用于测试 `my.onUserCaptureScreen` 截屏事件监听端口。

内存警告：模拟内存不足情况，用于测试 `my.onMemoryWarning` 内存警告监听端口。

mtop 环境切换：可进行环境切换，切换为 **日常、预发、线上**。

# Chapter 3

## 第三章 支付宝小程序开发基础知识



# 支付宝小程序代码结构及组成

## 了解小程序文件结构

本节以 IDE 内的 Todo App 模板小程序为例，介绍支付宝小程序的文件结构，以及每种文件类型在小程序中的作用。

Todo App 是一个简单的待办事项管理小程序，实现了用户登录、新增自定义待办事项、划除或恢复待办事项的功能。

### app.json

app.json 是小程序的全局配置，用于配置小程序的页面列表、默认窗口标题、导航栏背景色等。更多配置请参见 [文档配置](#)。

app.acss 定义了全局样式，作用于当前小程序的所有页面。

上例中的 page 为框架支持的特殊选择器，会匹配框架提供的页面根节点容器。

### app.js

app.js 用于注册小程序应用，可配置小程序的生命周期，声明全局数据，调用丰富的 API，如以下获取用户授权及获取用户信息 API 等，更多 API 信息请参见 [API 文档](#)。

可以看到，全局的逻辑代码放在 App({}) 中，声明了全局数据 todos 、userInfo，以及全局方法 getUserInfo()。

todos 全局数据中已经存储了一些数据，即 Todo App 小程序中已有的一些待办事项。

全局方法 getUserInfo() 调用了授权 API my.getAuthCode，以及获取用户信息 API my.getAuthUserInfo，并将获取到的用户信息存储在 userInfo 中。

## 小程序页面

此示例中有两个页面，Todo List 页面和 Add Todo 页面，都位于 pages 目录下。小程序的所有页面路径必须在 app.json 中申明，路径从项目根目录开始且不能包括后缀名，pages 的第一个页面就是小程序的首页。

每一个 [页面](#) 由同路径下的四种类型文件组成，即 .json 后缀的配置文件，.axml 后缀的模板文件，.acss 后缀的样式文件，.js 后缀的逻辑脚本文件。

### todo List 页面

todos.json 用于配置当前页面的窗口表现。此处定义了使用一个自定义组件 `add-button`，指定它的组件名称及对应的路径。自定义组件的具体使用后面会讲述。

页面配置文件不是必须的。当存在页面配置文件时，各个页面配置项会优先于 `app.json` 中 `window` 的同名配置项。当不存在页面配置文件，则直接使用 `app.json` 中的默认配置。因此，Todo List 页面的标题为 `app.json` 中指定的 `defaultTitle`，即 `Todo App`。

`todos.xml` 为页面结构模板文件。使用

`<view/>`, `<image/>`, `<text/>`, `<button/>`, `<label/>`, `<checkbox/>`,

来搭建页面结构以及通过 Mustache 语法两对大括号 (`{{}}`) 绑定 `todos` 数据。

- 绑定数据请参见[此文档](#)
- 绑定事件请参见[此文档](#)

`todos.js` 是页面的逻辑脚本文件，小程序页面的逻辑代码必需包含在 `Page({})` 中。在这个文件中可实现：

- 监听并处理页面的生命周期函数 `onShow` `onLoad`
- 获取小程序实例以及其他页面实例 `getApp` `getCurrentPages`
- 声明并处理数据 `data`
- 响应页面交互事件，调用 API 等
- 这里需要注意的是 `app.todos` 是来自 `app.js` 中全局的变量定义

`todos.acss` 定义页面局部样式。指定 `todos.xml` 中不同元素的样式，包括位置、背景颜色、字体、字体颜色等。ACSS 语法参见 [样式](#) 文档。页面的 `.acss` 文件不是必须的，但对于相同选择器，页面局部样式会覆盖 `app.acss` 全局样式。

## Add Todo 页面

`add-todo.json` 声明自定义组件名称和路径。

`add-todo.xml` 为页面结构模板文件。

此页面的两个核心功能为：

1. 使用 `<input/>` 组件接收用户输入。
2. `<add-button>` 是一个[自定义组件](#)，可将一些功能完整的代码封装为自定义组件，便于在其他地方复用。

`add-todo.js` 为页面逻辑代码。`add-todo.acss` 同 `todos.acss` 用法一致，不再赘述。



# 小程序全局结构

## 概述

App() 代表顶层应用，管理所有页面和全局数据，以及提供生命周期回调等。它也是一个构造方法，生成 App 实例。

一个小程序就是一个 App 实例。

每个小程序顶层一般包含三个文件。

- app.json：应用配置
- app.js：应用逻辑
- app.acss：应用样式（可选）

## 简单示例

一个简单的 app.json 代码如下：

```
{
  "pages": [
    "pages/index/index",
    "pages/logs/logs"
  ],
  "window": {
    "defaultTitle": "Demo"
  }
}
```

这段代码配置指定小程序包含两个页面（index 和 logs），以及应用窗口的默认标题设置为 “Demo”。

一个简单的 app.js 代码如下：

```
App({
  onLaunch(options) {
    // 第一次打开
  },
  onShow(options) {
    // 小程序启动，或从后台被重新打开
  },
  onHide() {
    // 小程序从前台进入后台
  },
}
```

```
onError(msg) {  
  // 小程序发生脚本错误或 API 调用出现报错  
  console.log(msg);  
},  
globalData: {  
  // 全局数据  
  name: 'alipay',  
},  
});
```

# app.json 全局配置

app.json 用于对小程序进行全局配置，设置页面文件的路径、窗口表现、多 tab 等。

以下是一个基本配置示例：

```
{
  "pages": [
    "pages/index/index",
    "pages/logs/index"
  ],
  "window": {
    "defaultTitle": "Demo"
  }
}
```

完整配置项如下：

| 属性     | 类型     | 是否必填 | 描述              |
|--------|--------|------|-----------------|
| pages  | Array  | 是    | 设置页面路径          |
| window | Object | 否    | 设置默认页面的窗口表现     |
| tabBar | Object | 否    | 设置底部 tabBar 的表现 |

## pages

app.json 中的 pages 为数组属性，数组中每一项都是字符串，用于指定小程序的页面。在小程序中新增或删除页面，都需要对 pages 数组进行修改。

pages 数组的每一项代表对应页面的路径信息，其中，第一项代表小程序的首页。

页面路径不需要写任何后缀，框架会自动去加载同名的 .json、.js、.axml、.acss 文件。举例来说，如果开发目录为：

```
├── pages
│   ├── index
│   │   ├── index.json
│   │   ├── index.js
│   │   ├── index.axml
│   │   └── index.acss
```

```

|   |——logs
|   |   |——logs.json
|   |   |——logs.js
|   |   |——logs.axml
|——app.json
|——app.js
|——app.acss

```

app.json 中应当如下配置：

```

{
  "pages": [
    "pages/index/index",
    "pages/logs/logs"
  ]
}

```

## window

window 用于设置小程序的状态栏、导航条、标题、窗口背景色等。

示例代码：

```

{
  "window": {
    "defaultTitle": "支付宝接口功能演示"
  }
}

```

| 属性                   | 类型     | 是否必填 | 描述   | 最低版本 |
|----------------------|--------|------|--|------|
| defaultTitle         | String | 否    | 页面默认标题   | -    |
| pullRefresh          | String | 否    | 是否允许下拉刷新。默认 NO，备注：下拉刷新生效的前提是 allowsBounceVertical 值为 YES | -    |
| allowsBounceVertical | String | 否    | 是否允许向下拉拽。默认 YES，支持 YES / NO                              | -    |

|                       |          |   |  |  |
|-----------------------|----------|---|--|--|
| transparentTitle      | String   | 否 | 导航栏透明设置。默认 none，支持 always 一直透明 / auto 滑动自适应 / none 不透明   | -  |
| titlePenetrate        | String   | 否 | 是否允许导航栏点击穿透。默认 NO，支持 YES / NO                            | -  |
| showTitleLoading      | String   | 否 | 是否进入时显示导航栏的 loading。默认 NO，支持 YES / NO                    | -  |
| titleImage            | String   | 否 | 导航栏图片地址  | -  |
| titleBarColor         | HexColor | 否 | 导航栏背景色，十六进制颜色值（0-255）                                    | -  |
| backgroundColor       | HexColor | 否 | 页面的背景色，十六进制颜色值（0-255）                                    | -  |
| backgroundImageColor  | HexColor | 否 | 下拉露出显示的背景图底色，十六进制颜色值（0-255）                              | -  |
| backgroundImageUrl    | String   | 否 | 下拉露出显示的背景图链接   | -  |
| gestureBack           | String   | 否 | iOS 用，是否支持手势返回。默认 NO，支持 YES / NO                         | -  |
| enableScrollbar       | Boolean  | 否 | Android 用，是否显示 WebView 滚动条。默认 YES，支持 YES / NO            | -  |
| onReachBottomDistance | Number   | 否 | 页面上拉触底时触发时距离页面底部的距离，单位为 px。相关文档 <a href="#">页面事件处理函数</a> | <a href="#">1.19.0</a> ，目前 iOS 在 page.json 下设置无效，只能全局设置。 |

## tabBar

如果你的小程序是一个多 tab 应用（客户端窗口的底部栏可以切换页面），那么可以通过 tabBar 配置项指定 tab 栏的表现，以及 tab 切换时显示的对应页面。

注意：

- 通过页面跳转（[my.navigateTo](#)）或者页面重定向（[my.redirectTo](#)）所到达的页面，即使它是定义在 tabBar 配置中的页面，也不会显示底部的 tab 栏。
- tabBar 的第一个页面必须是首页。

tabBar 配置项有以下：

| 属性              | 类型       | 是否必填 | 描述        |
|-----------------|----------|------|-----------|
| textColor       | HexColor | 否    | 文字颜色      |
| selectedColor   | HexColor | 否    | 选中文字颜色    |
| backgroundColor | HexColor | 否    | 背景色       |
| items           | Array    | 是    | 每个 tab 配置 |

每个 item 配置：

| 属性         | 类型     | 是否必填 | 描述     |
|------------|--------|------|--------|
| pagePath   | String | 是    | 设置页面路径 |
| name       | String | 是    | 名称     |
| icon       | String | 否    | 平常图标路径 |
| activeIcon | String | 否    | 高亮图标路径 |

icon 图标推荐大小为 60×60 px 大小，系统会对传入的非推荐尺寸的图片进行非等比拉伸或缩放。

示例代码：

```
{  
  "tabBar": {
```

```
"textColor": "#ddddd",
"selectedColor": "#49a9ee",
"backgroundColor": "#ffffff",
"items": [
  {
    "pagePath": "pages/index/index",
    "name": "首页"
  },
  {
    "pagePath": "pages/logs/logs",
    "name": "日志"
  }
]
}
```

## app.acss 全局样式

app.acss 作为全局样式，作用于当前小程序的所有页面。

ACSS 是一套样式语言，用于描述 AXML 的组件样式，决定 AXML 的组件的显示效果。

为适应广大前端开发者，ACSS 和 CSS 规则完全一致，100% 可以用。同时为更适合开发小程序，对 CSS 进行了扩充。

ACSS 支持 px, rpx, vh, vw 等单位。

### rpx

rpx (responsive pixel) 可以根据屏幕宽度进行自适应，规定屏幕宽为 750rpx。以 Apple iPhone6 为例，屏幕宽度为 375px，共有 750 个物理像素，则  $750\text{rpx} = 375\text{px} = 750$  物理像素， $1\text{rpx} = 0.5\text{px} = 1$  物理像素。

| 设备           | rpx 换算 px (屏幕宽度 / 750)         | px 换算 rpx (750 / 屏幕宽度)        |
|--------------|--------------------------------|-------------------------------|
| iPhone5      | $1\text{rpx} = 0.42\text{px}$  | $1\text{px} = 2.34\text{rpx}$ |
| iPhone6      | $1\text{rpx} = 0.5\text{px}$   | $1\text{px} = 2\text{rpx}$    |
| iPhone6 Plus | $1\text{rpx} = 0.552\text{px}$ | $1\text{px} = 1.81\text{rpx}$ |

### 样式导入

使用 @import 语句可以导入外联样式表，@import 后需要加上外联样式表相对路径，用 ; 表示结束。

示例代码：

```
/** button.acss */
.sm-button {
  padding: 5px;
}
```

```
/** app.acss */
@import "../button.acss";
.md-button {
  padding: 15px;
}
```

导入路径支持从 node\_modules 目录载入第三方模块，例如 page.acss:

```
@import "../button.acss"; /*相对路径*/
```



```
@import "/button.acss"; /*项目绝对路径*/
@import "third-party/page.acss"; /*第三方 npm 包路径*/
```

## 内联样式

组件上支持使用 `style`、`class` 属性来控制样式。

### style 属性

用于接收动态样式，样式在运行时会进行解析。行内样式不支持 `!important` 优先级规则。

```
<view style="color:{{color}};" />
```

### class 属性

用于接收静态样式，属性值是样式规则中类选择器名（样式类名）的集合，样式类名不需要带上 `.`，多个类名间以空格分隔。请静态样式写进 `class` 中，避免将静态样式写进 `style` 中，以免影响渲染速度。

```
<view class="my-awesome-view" />
```

## 选择器

同 CSS3 保持一致。

注意：

- `.a-`、`.am-` 开头的类选择器为系统组件占用，不可使用。
- 不支持属性选择器。

## 全局样式与局部样式

- `app.acss` 中的样式为全局样式，作用于每一个页面。
- 页面文件夹内的 `.acss` 文件中定义的样式为局部样式，只作用在对应的页面，并会覆盖 `app.acss` 中相同的选择器。

## 本地资源引用

ACSS 文件里的本地资源引用请使用绝对路径的方式，不支持相对路径引用。例如：

```
/* 支持 */
background-image: url('/images/ant.png');
/* 不支持 */
background-image: url('./images/ant.png');
```

## app.js 注册小程序

### App(object: Object)

App() 用于注册小程序，接受一个 Object 作为属性，用来配置小程序的生命周期等。

App() 必须在 app.js 中调用，必须调用且只能调用一次。

### object 属性说明

| 属性                | 类型       | 描述              | 触发时机                           |
|-------------------|----------|-----------------|--------------------------------|
| onLaunch          | Function | 生命周期回调：监听小程序初始化 | 当小程序初始化完成时触发，全局只触发一次           |
| onShow            | Function | 生命周期回调：监听小程序显示  | 当小程序启动，或从后台进入前台显示时触发           |
| onHide            | Function | 生命周期回调：监听小程序隐藏  | 当当前页面被隐藏时触发，例如跳转、按下设备 Home 键离开 |
| onError           | Function | 监听小程序错误         | 当小程序发生 js 错误时触发                |
| onShareAppMessage | Function | 全局分享配置          | -                              |

前台/后台定义：

- 小程序用户点击右上角关闭，或者按下设备 Home 键离开支付宝时，小程序并不会直接销毁，而是进入后台。
- 当用户再次进入支付宝或再次打开小程序时，小程序会从后台进入前台。
- 只有当小程序进入后台 5 分钟后，或占用系统资源过高，才会被真正销毁。

### onLaunch(object: Object) 及 onShow(object: Object)

object 属性说明：

| 属性 | 类型 | 描述 |
|----|----|----|
|----|----|----|

|              |        |  |
|--------------|--------|--|
| query        | Object | 当前小程序的 query，从启动参数的 query 字段解析而来           |
| scene        | number | 启动小程序的 <a href="#">场景值</a>                 |
| path         | string | 当前小程序的页面地址，从启动参数 page 字段解析而来，page 忽略时默认为首页 |
| referrerInfo | Object | 来源信息                                       |

比如，启动小程序的 schema url 如下：

```
alipays://platformapi/startapp?appId=1999&query=number%3D1&page=x%2Fy%2Fz
```

- 小程序首次启动时，onLaunch 方法可获取 query、path 属性值。
- 小程序在后台被用 schema 打开，也可从 onShow 方法中获取 query、path 属性值。

```
App({
  onLaunch(options) {
    // 第一次打开
    console.log(options.query);
    // {number:1}
    console.log(options.path);
    // x/y/z
  },
  onShow(options) {
    // 从后台被 schema 重新打开
    console.log(options.query);
    // {number:1}
    console.log(options.path);
    // x/y/z
  },
});
```

referrerInfo 子属性说明：

| 属性              | 类型     | 描述                | 最低版本                   |
|-----------------|--------|-------------------|------------------------|
| appId           | string | 来源小程序             | -                      |
| sourceServiceId | string | 来源插件，当处于插件运行模式时可见 | <a href="#">1.11.0</a> |

|           |        |              |   |
|-----------|--------|--------------|---|
| extraData | Object | 来源小程序传过来的数据。 | - |
|-----------|--------|--------------|---|

注意：

- 不要在 onShow 中进行 redirectTo 或 navigateTo 等操作页面栈的行为。
- 不要在 onLaunch 里调用 [getCurrentPages\(\)](#)，因为此时 page 还未生成。

## onHide()

小程序从前台进入后台时触发 onHide() 。

示例代码：

```
App({
  onHide() {
    // 进入后台时
    console.log('app hide');
  },
});
```

## onError(error: String)

小程序发生脚本错误时触发。

示例代码：

```
App({
  onError(error) {
    // 小程序执行出错时
    console.log(error);
  },
});
```

## onShareAppMessage(object: Object)

全局分享配置。当页面未设置 page.onShareAppMessage 时，调用分享会执行全局的分享设置，具体见 [分享](#) 。

## globalData 全局数据

App() 中可以设置全局数据 globalData。

示例代码：

```
// app.js
App({
  globalData: 1
});
```

## getApp 方法

小程序提供了全局的 `getApp()` 方法，可获取当前小程序实例，一般用于在子页面中获取顶层应用。

```
var app = getApp();  
console.log(app.globalData); // 获取 globalData
```

使用过程中，请注意以下几点：

- `App()` 函数中不可以调用 `getApp()`，可使用 `this` 可以获取当前小程序实例。
- 通过 `getApp()` 获取实例后，请勿私自调用生命周期回调函数。
- 请区分全局变量及页面局部变量，比如：

```
// app.js  
App({  
  //定义全局变量 globalData，在整个 App 中有效  
  globalData: 1  
});
```

```
// a.js  
// 定义页面局部变量 localValue，只在 a.js 有效  
var localValue = 'a';  
// 获取 app 实例  
var app = getApp();  
// 拿到全局数据，并改变它  
app.globalData++;
```

```
// b.js  
// 定义页面局部变量 localValue，只在 b.js 有效  
var localValue = 'b';  
// 如果 a.js 先运行，globalData 会返回 2  
console.log(getApp().globalData);
```

`a.js` 和 `b.js` 两个文件中都声明了变量 `localValue`，但并不会互相影响，因为各个文件声明的局部变量和函数只在当前文件下有效。

# 小程序页面结构

## 概述

**Page** 代表应用的一个页面，负责页面展示和交互。每个页面对应一个子目录，一般有多少个页面，就有多少个子目录。它也是一个构造函数，用来生成页面实例。每个小程序页面一般包含四个文件。

- [pageName].js: 页面逻辑
- [pageName].xml: 页面结构
- [pageName].acss: 页面样式（可选）
- [pageName].json: 页面配置（可选）

页面初始化时，提供数据。

```
Page({
  data: {
    title: 'Alipay',
    array: [{user: 'li'}, {user: 'zhao'}],
  },
});
```

根据以上提供的数据，渲染页面内容。

```
<view>{{title}}</view>
<view>{{array[0].user}}</view>
```

定义交互行为时，需要指定响应函数。

```
<view onTap="handleTap">click me</view>
```

以上代码指定用户触摸按钮时，调用 `handleTap` 方法。

```
Page({
  handleTap() {
    console.log('yo! view tap!');
  },
});
```

页面重新渲染，需要在页面脚本里面调用 `this.setData` 方法。

```
<view>{{text}}</view>
<button onTap="changeText">Change normal data </button>
```

以上代码指定用户触摸按钮时，调用 `changeText` 方法。

```
Page({
  data: {
    text: 'init data',
  },
});
```

```
},  
changeText() {  
  this.setData({  
    text: 'changed data',  
  });  
},  
});
```

上面代码中，changeText 方法里面调用 this.setData 方法，会导致页面重新渲染。

## 页面配置

在 /pages 目录中的 .json 文件用于配置当前页面的窗口表现。页面配置比 app.json 全局配置简单得多，只能设置 window 相关配置项，但无需写 window 这个键。页面配置项会优先于全局配置项。

window 配置项同 [全局配置项](#)，同时支持以下几点：

- 支持 optionMenu 配置导航图标，点击后触发 onOptionMenuClick。但注意：optionMenu 配置将被废弃，建议使用 my.setOptionMenu 设置导航栏图标。
- 支持 titlePenetrate，设置导航栏点击穿透。
- 支持 barButtonTheme，设置导航栏图标主题。

完整配置项如下：

| 属性名            | 类型     | 必填 | 描述  | 最低版本           |
|----------------|--------|----|---|----------------|
| optionMenu     | Object | 否  | 设置导航栏额外图标，目前支持设置属性 icon，值为图标 url（以 https/http 开头）或 base64 字符串，大小建议 30*30 px | 基础库 1.3.0      |
| titlePenetrate | BOOL   | 否  | 设置导航栏点击穿透   | 支付宝客户端 10.1.52 |
| barButtonTheme | String | 否  | 设置导航栏图标主题。“default”为蓝色图标，“light”为白色图标                                       | 支付宝客户端 10.1.52 |

以下为一个基本示例：

```
{
  "optionMenu": {
    "icon": "https://img.alicdn.com/tps/i3/T1OjaVfL4dXXa.JOZB-114-114.png"
  },
  "titlePenetrate": "YES",
  "barButtonTheme": "light"
}
```



## 页面结构

在 /pages 目录中的 .axml 文件用于定义当前这个页面的结构，文件内容遵循 AXML 语法。

AXML 和 HTML 非常相似，但是也有很多不一样的地方。

AXML 是小程序框架设计的一套标签语言，用于描述小程序页面的结构。

AXML 语法可分为五个部分：[数据绑定](#)、[条件渲染](#)、[列表渲染](#)、[模板](#)、[引用](#)。

AXML 代码示例：

```
<!-- pages/index/index.axml -->
<view a:for="{{items}}"> {{item}} </view>
<view a:if="{{view == 'WEBVIEW'}}"> WEBVIEW </view>
<view a:elif="{{view == 'APP'}}"> APP </view>
<view a:else> alipay </view>
<view onTap="add"> {{count}} </view>
```

对应的 .js 文件示例：

```
// pages/index/index.js
Page({
  data: {
    items: [1, 2, 3, 4, 5, 6, 7],
    view: 'alipay',
    count: 1,
  },
  add(e) {
    this.setData({
      count: this.data.count + 1,
    });
  },
});
```

对应的 .acss 文件示例：

```
/* pages/index/index.acss */
view {
  padding-left: 10px;
}
```

## 页面样式

在 /pages 目录中的 .acss 文件用于定义页面样式。

每个页面中的根元素为 page，需要设置页面高度或背景色时，可按如下方式：

```
page {  
  background-color: #fff;  
}
```

ACSS 是一套样式语言，用于描述 AXML 的组件样式，决定 AXML 的组件的显示效果。

为适应广大前端开发者，ACSS 和 CSS 规则完全一致，100% 可以用。同时为更适合开发小程序，对 CSS 进行了扩充。

ACSS 支持 px, rpx, vh, vw 等单位。

### rpx

rpx (responsive pixel) 可以根据屏幕宽度进行自适应，规定屏幕宽为 750rpx。以 Apple iPhone6 为例，屏幕宽度为 375px，共有 750 个物理像素，则  $750rpx = 375px = 750$  物理像素， $1rpx = 0.5px = 1$  物理像素。

| 设备           | rpx 换算 px (屏幕宽度 / 750) | px 换算 rpx (750 / 屏幕宽度) |
|--------------|------------------------|------------------------|
| iPhone5      | 1rpx = 0.42px          | 1px = 2.34rpx          |
| iPhone6      | 1rpx = 0.5px           | 1px = 2rpx             |
| iPhone6 Plus | 1rpx = 0.552px         | 1px = 1.81rpx          |

## 样式导入

使用 @import 语句可以导入外联样式表，@import 后需要加上外联样式表相对路径，用 ; 表示结束。

示例代码：

```
/** button.acss */  
.sm-button {  
  padding: 5px;  
}
```

```
/** app.acss */  
@import "../button.acss";
```

```
.md-button {  
  padding: 15px;  
}
```

导入路径支持从 node\_modules 目录载入第三方模块，例如 page.acss:

```
@import "../button.acss"; /*相对路径*/  
@import "/button.acss"; /*项目绝对路径*/  
@import "third-party/page.acss"; /*第三方 npm 包路径*/
```

## 内联样式

组件上支持使用 style、class 属性来控制样式。

### style 属性

用于接收动态样式，样式在运行时会进行解析。行内样式不支持 !important 优先级规则。

```
<view style="color:{{color}};" />
```

### class 属性

用于接收静态样式，属性值是样式规则中类选择器名（样式类名）的集合，样式类名不需要带上.，多个类名间以空格分隔。请静态样式写进 class 中，避免将静态样式写进 style 中，以免影响渲染速度。

```
<view class="my-awesome-view" />
```

## 选择器

同 CSS3 保持一致。

注意：

- .a-, .am- 开头的类选择器为系统组件占用，不可使用。
- 不支持属性选择器。

## 全局样式与局部样式

- app.acss 中的样式为全局样式，作用于每一个页面。
- 页面文件夹内的 .acss 文件中定义的样式为局部样式，只作用在对应的页面，并会覆盖 app.acss 中相同的选择器。

## 本地资源引用

ACSS 文件里的本地资源引用请使用绝对路径的方式，不支持相对路径引用。例如：

```
/* 支持 */  
background-image: url('/images/ant.png');  
/* 不支持 */  
background-image: url('./images/ant.png');
```

## 页面运行机制

### Page(object: Object)

在 /pages 目录的 .js 文件中，定义 Page()，用于注册一个小程序页面，接受一个 object 作为属性，用来指定页面的初始数据、生命周期回调、事件处理等。以下为一个基本的页面代码：

```
// pages/index/index.js
Page({
  data: {
    title: "Alipay",
  },
  onLoad(query) {
    // 页面加载
  },
  onShow() {
    // 页面显示
  },
  onReady() {
    // 页面加载完成
  },
  onHide() {
    // 页面隐藏
  },
  onUnload() {
    // 页面被关闭
  },
  onTitleClick() {
    // 标题被点击
  },
  onPullDownRefresh() {
    // 页面被下拉
  },
  onReachBottom() {
    // 页面被拉到底部
  },
  onShareAppMessage() {
```

```

    // 返回自定义分享信息
  },
  // 事件处理函数对象
  events: {
    onBack() {
      console.log('onBack');
    },
  },
  // 自定义事件处理函数
  viewTap() {
    this.setData({
      text: 'Set data for update.',
    });
  },
  // 自定义事件处理函数
  go() {
    // 带参数的跳转，从 page/ui/index 的 onLoad 函数的 query 中读取 type
    my.navigateTo({url: '/page/ui/index?type=mini'});
  },
  // 自定义数据对象
  customData: {
    name: 'alipay',
  },
});

```

## 页面生命周期

下图说明了页面 Page 对象的生命周期。

小程序主要靠视图线程（Webview）和应用服务线程（Worker）来控制管理。视图线程和应用服务线程同时运行。

- 应用服务线程启动后运行 `app.onLaunch` 和 `app.onShow` 以完成 App 创建，再运行 `page.onLoad` 和 `page.onShow` 以完成 Page 创建，此时等待视图线程初始化完成通知。
- 视图线程初始化完成通知应用服务线程，应用服务线程将初始化数据发送给视图线程进行渲染，此时视图线程完成第一次数据渲染。
- 第一次渲染完成后视图线程进入就绪状态并通知应用服务线程，应用服务线程调用 `page.onReady` 函数并进入活动状态。

- 应用线程进入活动状态后每次数据修改将会通知视图线程进行渲染。当切换页面进入后台，应用线程调用 `page.onHide` 函数后，进入存活状态；页面返回到前台将调用 `page.onShow` 函数，进入活动状态；当调用返回或重定向页面后将调用 `page.onUnload` 函数，进行页面销毁。



## object 属性说明

| 属性     | 类型                      | 描述               | 最低版本                   |
|--------|-------------------------|------------------|------------------------|
| data   | Object   Function       | 初始数据或返回初始化数据的函数。 | -                      |
| events | Object                  | 事件处理函数对象         | <a href="#">1.13.7</a> |
| onLoad | Function(query: Object) | 页面加载时触发          | -                      |

|                   |                               |   |                        |
|-------------------|-------------------------------|---|------------------------|
| onShow            | Function                      | 页面显示时触发                                       | -                      |
| onReady           | Function                      | 页面初次渲染完成时触发                                   | -                      |
| onHide            | Function                      | 页面隐藏时触发                                       | -                      |
| onUnload          | Function                      | 页面卸载时触发                                       | -                      |
| onShareAppMessage | Function(options: Object)     | 点击右上角分享时触发                                    | -                      |
| onTitleClick      | Function                      | 点击标题触发  | -                      |
| onOptionMenuClick | Function                      | 点击导航栏额外图标触发                                   | <a href="#">1.3.0</a>  |
| onPopMenuClick    | Function                      | 点击右上角通用菜单中的自定义菜单按钮触发                          | <a href="#">1.3.0</a>  |
| onPullDownRefresh | Function({from: manual code}) | 页面下拉时触发                                       | -                      |
| onPullIntercept   | Function                      | 下拉中断时触发                                       | <a href="#">1.11.0</a> |
| onTabItemTap      | Function                      | 点击 tabItem 时触发                                | <a href="#">1.11.0</a> |
| onPageScroll      | Function({scrollTop})         | 页面滚动时触发                                       | -                      |
| onReachBottom     | Function                      | 上拉触底时触发                                       | -                      |
| 其他                | Any                           | 开发者可以添加任意的函数或属性到 object 中，在页面的函数中可以用 this 来访问 | -                      |

## 页面数据对象 data

通过设置 data 指定页面的初始数据。当 data 为对象时，被所有页面共享。



即：当该页面回退后再次进入该页面时，会显示上次页面的数据，而非初始数据。这种情况，可以通过设置 data 为不可变数据或者变更 data 为页面独有数据两种方式来解决。

### 设置为不可变数据

```
Page({
  data: { arr:[] },
  dolt() {
    this.setData({arr: [...this.data.arr, 1]});
  },
});
```

### 设置页面独有数据（不推荐）

```
Page({
  data() { return { arr:[] }; },
  dolt() {
    this.setData({arr: [1, 2, 3]});
  },
});
```

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

比如：

```
Page({
  data: { arr:[] },
  dolt() {
    this.data.arr.push(1); // 不要这么写！
    this.setData({arr: this.data.arr});
  }
});
```

## 生命周期函数

### onLoad(query: Object)

页面初始化时触发。一个页面只会调用一次。

query 为 my.navigateTo 和 my.redirectTo 中传递的 query 对象。

query 内容格式为：“参数名=参数值&参数名=参数值...”。

| 属性    | 类型     | 描述           |
|-------|--------|--------------|
| query | Object | 打开当前页面路径中的参数 |

### onShow()

页面显示/切入前台时触发。

### onReady()

页面初次渲染完成时触发。

一个页面只会调用一次，代表页面已经准备妥当，可以和视图层进行交互。

对界面的设置，如 my.setNavigationBar 请在 onReady 之后设置。

### onHide()

页面隐藏/切入后台时触发。

如 my.navigateTo 到其他页面或底部 tab 切换等。

### onUnload()

页面卸载时触发。

如 my.redirectTo 或 my.navigateBack 到其他页面等。

## 页面事件处理函数

### onShareAppMessage(options: Object)

点击右上角通用菜单中的分享按钮时或点击页面内分享按钮时触发。详见[分享](#)。

### onTitleClick()

点击标题时触发。

### onOptionMenuClick()

点击右上角菜单按钮时触发。

### onPopMenuClick()

点击右上角通用菜单按钮时触发。

### onPullDownRefresh({from: manual|code})

下拉刷新时触发。

需要先在 [app.json](#) 的 window 选项中开启 pullRefresh 。当处理完数据刷新后，my.stopPullDownRefresh 可以停止当前页面的下拉刷新。

### onPullIntercept()

下拉中断时触发。

### onTabItemTap(object: Object)

点击 tabItem 时触发。

| 属性       | 类型     | 描述                     |
|----------|--------|------------------------|
| from     | String | 点击来源                   |
| pagePath | String | 被点击 tabItem 的页面路径      |
| text     | String | 被点击 tabItem 的按钮文字      |
| index    | Number | 被点击 tabItem 的序号，从 0 开始 |

### onPageScroll({scrollTop})

页面滚动时触发，scrollTop 为页面滚动距离。

## onReachBottom()

上拉触底时触发。

## events

为了使代码更加简洁，提供了新的事件处理对象 `events`。

已有的页面处理事件函数跟直接在 `page` 实例上暴露的事件函数等价。

注意：

- `events` 从基础库 1.13.7 版本 开始支持。
- 请正确区分页面事件函数与 `events` 内同名事件函数的基础库版本要求。

以下是 `events` 支持的事件函数列表：

| 事件                             | 类型  | 描述                             | 最低版本                   |
|--------------------------------|---|--------------------------------|------------------------|
| <code>onBack</code>            | Function  | 页面返回时触发                        | <a href="#">1.13.7</a> |
| <code>onKeyboardHeight</code>  | Function  | 键盘高度变化时触发                      | <a href="#">1.13.7</a> |
| <code>onOptionMenuClick</code> | Function  | 点击右上角菜单按钮触发                    | <a href="#">1.13.7</a> |
| <code>onPopMenuClick</code>    | Function  | 点击右上角通用菜单按钮触发                  | <a href="#">1.13.7</a> |
| <code>onPullIntercept</code>   | Function  | 下拉截断时触发                        | <a href="#">1.13.7</a> |
| <code>onPullDownRefresh</code> | Function({from: manual/code})                                 | 页面下拉时触发                        | <a href="#">1.13.7</a> |
| <code>onTitleClick</code>      | Function  | 点击标题触发                         | <a href="#">1.13.7</a> |
| <code>onTabItemTap</code>      | Function  | 点击非当前 <code>tabItem</code> 后触发 | <a href="#">1.13.7</a> |
| <code>beforeTabItemTap</code>  | Function  | 点击非当前 <code>tabItem</code> 前触发 | <a href="#">1.13.7</a> |
| <code>onResize</code>          | Function({size: {windowWidth: number, windowHeight: number}}) | window 尺寸改变时触发                 | <a href="#">1.16.0</a> |

示例代码：

```
// 特征检测
my.canIUse('page.events.onBack');

Page({
  data: {
    text: 'This is page data.'
  },
  onLoad(){
    // 页面加载时触发
  },
  events:{
    onBack(){
      // 页面返回时触发
    },
    onKeyboardHeight(e){
      // 键盘高度变化时触发
      console.log('键盘高度: ', e.height)
    },
    onOptionMenuClick(){
      // 点击右上角菜单按钮触发
    },
    onPopMenuClick(e){
      // 点击右上角通用菜单中的自定义菜单按钮触发
      console.log('用户点击自定义菜单的索引', e.index)
      console.log('用户点击自定义菜单的 name', e.name)
      console.log('用户点击自定义菜单的 menuIconUrl', e.menuIconUrl)
    },
    onPullIntercept(){
      // 下拉截断时触发
    },
    onPullDownRefresh(e){
      // 页面下拉时触发。e.from 的值是 “code” 表示 startPullDownRefresh 触发的事件；值是 “manual” 表示用户下拉触发的下拉事件
      console.log('触发下拉刷新的类型', e.from)
      my.stopPullDownRefresh()
    },
  },
}
```

```

onTitleClick(){
    // 点击标题触发
},
onTabItemTap(e){
    // e.from 是点击且切换 tabItem 后触发，值是 “user” 表示用户点击触发的事件；
    // 值是 “api” 表示 switchTab 触发的事件
    console.log('触发 tab 变化的类型', e.from)
    console.log('点击的 tab 对应页面的路径', e.pagePath)
    console.log('点击的 tab 的文字', e.text)
    console.log('点击的 tab 的索引', e.index)
},
beforeTabItemTap(){
    // 点击但切换 tabItem 前触发
},
onResize(e){
    // window 尺寸改变时触发
    var {windowWidth, windowHeight} = e.size
    console.log('改变后 window 的宽度', windowWidth)
    console.log('改变后 window 的高度', windowHeight)
},
}
})

```

## Page.prototype.setData(data: Object, callback: Function)

setData 会将数据从逻辑层发送到视图层，同时改变对应的 this.data 的值。

Object 以 key: value 的形式表示，将 this.data 中的 key 对应的值改变成 value。其中 key 可以非常灵活，以数据路径的形式给出，如 array[2].message、a.b.c.d，可以不需要在 this.data 中预先定义。

使用过程中，需要注意以下几点：

1. 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。
2. 仅支持设置可 JSON 化的数据。
3. 请尽量避免一次设置过多的数据。
4. 请不要把 data 中任何一项的 value 设为 undefined，否则这一项将不被设置并可能遗留一些潜在问题。

示例代码：

```

<view>{{text}}</view>
<button onTap="changeTitle"> Change normal data </button>
<view>{{array[0].text}}</view>
<button onTap="changeArray"> Change Array data </button>
<view>{{object.text}}</view>
<button onTap="changePlanetColor"> Change Object data </button>
<view>{{newField.text}}</view>
<button onTap="addNewKey"> Add new data </button>
<view>hello: {{name}}</view>
<button onTap="changeName"> Change name </button>

```

```

Page({
  data: {
    text: 'test',
    array: [{text: 'a'}],
    object: {
      text: 'blue',
    },
    name: 'taobao',
  },
  changeTitle() {
    // 错误！不要直接去修改 data 里的数据
    // this.data.text = 'changed data'

    // 正确
    this.setData({
      text: 'ha',
    });
  },
  changeArray() {
    // 可以直接使用数据路径来修改数据
    this.setData({
      'array[0].text': 'b',
    });
  },
  changePlanetColor(){
    this.setData({

```

```

        'object.text': 'red',
    });
},
addNewKey() {
    this.setData({
        'newField.text': 'c',
    });
},
changeName() {
    this.setData({
        name: 'alipay',
    }, () => { // 接受传递回调函数
        console.log(this); // this 当前页面实例
        this.setData({ name: this.data.name + ', ' + 'welcome!'});
    });
},
});

```

参数说明：

| 事件       | 类型       | 描述                  | 最低版本  |
|----------|----------|---------------------|---|
| data     | Object   | 待改变的数据              | -   |
| callback | Function | 回调函数，在页面渲染更新完成之后执行。 | 使用<br>my.canIUse('page.setData.callb<br>ack') 做兼容性处理。详见 <a href="#">1.7.0</a> |

## Page.prototype.\$spliceData(data: Object, callback: Function)

**说明：** \$spliceData 自 1.7.2 之后才支持，可以使用

**my.canIUse('page.\$spliceData')** 做兼容性处理。详见 [兼容接口说明](#)。

spliceData 同样用于将数据从逻辑层发送到视图层，但是相比于 setData，在处理长列表的时候，其具有更高的性能。

Object 以 key: value 的形式表示，将 this.data 中的 key 对应的值改变成 value。其中 key 可以非常灵活，以数据路径的形式给出，如 array[2].message、a.b.c.d，可以不需要在 this.data 中预先定义。value 为一个数组（格式：[start, deleteCount, ...items]），数组的第一个元素为操作的



起始位置，第二个元素为删除的元素的个数，剩余的元素均为插入的数据。对应 es5 中数组的 splice 方法。

示例代码：

```
<!-- pages/index/index.xml -->
<view class="spliceData">
  <view a:for="{{a.b}}" key="{{item}}" style="border:1px solid red">
    {{item}}
  </view>
</view>
```

```
// pages/index/index.js
Page({
  data: {
    a: {
      b: [1,2,3,4],
    },
  },
  onLoad(){
    this.$spliceData({ 'a.b': [1, 0, 5, 6] });
  },
});
```

页面输出：

```
1
5
6
2
3
4
```

参数说明：

| 事件       | 类型       | 描述                  |
|----------|----------|---------------------|
| data     | Object   | 待改变的数据              |
| callback | Function | 回调函数，在页面渲染更新完成之后执行。 |

## Page.prototype.\$batchedUpdates(callback: Function)

批量更新数据。

**说明** \$batchedUpdates 自 1.14.0 之后才支持，可以使用

my.canIUse('page.\$batchedUpdates') 做兼容性处理。详见 [兼容接口说明](#)

参数说明：

| 事件       | 类型       | 描述                  |
|----------|----------|---------------------|
| callback | Function | 在此回调函数中的数据操作会被批量更新。 |

下面是示例代码：

```
// pages/index/index.js
Page({
  data: {
    counter: 0,
  },
  plus() {
    setTimeout(() => {
      this.$batchedUpdates(() => {
        this.setData({
          counter: this.data.counter + 1,
        });
        this.setData({
          counter: this.data.counter + 1,
        });
      });
    }, 200);
  },
});
```

```
<!-- pages/index/index.xml -->
<view>{{counter}}</view>
<button onTap="plus">+2</button>
```

1. 本示例中每次点击按钮，页面的 counter 会加 2

2. 将 `setData` 放在 `this.$batchedUpdates` 中，这样尽管有多次 `setData`，但是却只有一次数据的传输

## Page.route

Page 路径，对应 `app.json` 中配置的路径值，类型为 `String`。  
这是一个只读属性。

```
Page({
  onShow() {
    // 对应 app.json 中配置的路径值
    console.log(this.route)
  }
})
```

## getCurrentPages 方法

getCurrentPages() 方法用于获取当前页面栈的实例，返回页面数组栈。第一个元素为首页，最后一个元素为当前页面。

框架以栈的形式维护当前的所有页面。路由切换与页面栈的关系如下：

| 路由方式   | 页面栈表现               |
|--------|---------------------|
| 初始化    | 新页面入栈               |
| 打开新页面  | 新页面入栈               |
| 页面重定向  | 当前页面出栈，新页面入栈        |
| 页面返回   | 当前页面出栈              |
| Tab 切换 | 页面全部出栈，只留下新的 Tab 页面 |

下面代码可以用于检测当前页面栈是否具有 5 层页面深度。

```
if (getCurrentPages().length === 5) {  
  my.redirectTo({  
    url: '/pages/logs/logs'  
  });  
} else {  
  my.navigateTo({  
    url: '/pages/index/index'  
  });  
}
```

**注意：** 不要尝试修改页面栈，会导致路由以及页面状态错误。

# 支付宝小程序 API

## 简介

支付宝小程序开发框架提供了丰富的 JSAPI（原生 API）和 OpenAPI（开放能力 API），开发者可方便快捷地调用这些 API，详情请参见 [API 概览](#)。

- OpenAPI 是支付宝开放平台在小程序上开放的开放能力 API。通过 OpenAPI，小程序可以轻松实现用户授权、获取会员基础信息、获取用户手机号、小程序唤起支付、跳转支付宝卡包、会员开卡授权等多种多样的功能。
- JSAPI 按实现的功能分类，可分为界面、多媒体、缓存、文件、位置、网络、设备、数据安全、分享、收藏、自定义通用菜单、小程序当前运行版本类型、自定义分析、更新管理等 14 个大类。

本章我们将讲述支付宝小程序比较常用的几个 JSAPI，带你走进 JSAPI 的奇妙世界。

# 如何秘密告白：小程序 HTTPS 网络请求实现

木心的《从前慢》里说：

记得早先少年时  
大家诚诚恳恳  
说一句 是一句  
.....  
从前的日色变得慢  
车，马，邮件都慢  
一生只够爱一个人  
从前的锁也好看  
钥匙精美有样子  
你锁了 人家就懂了

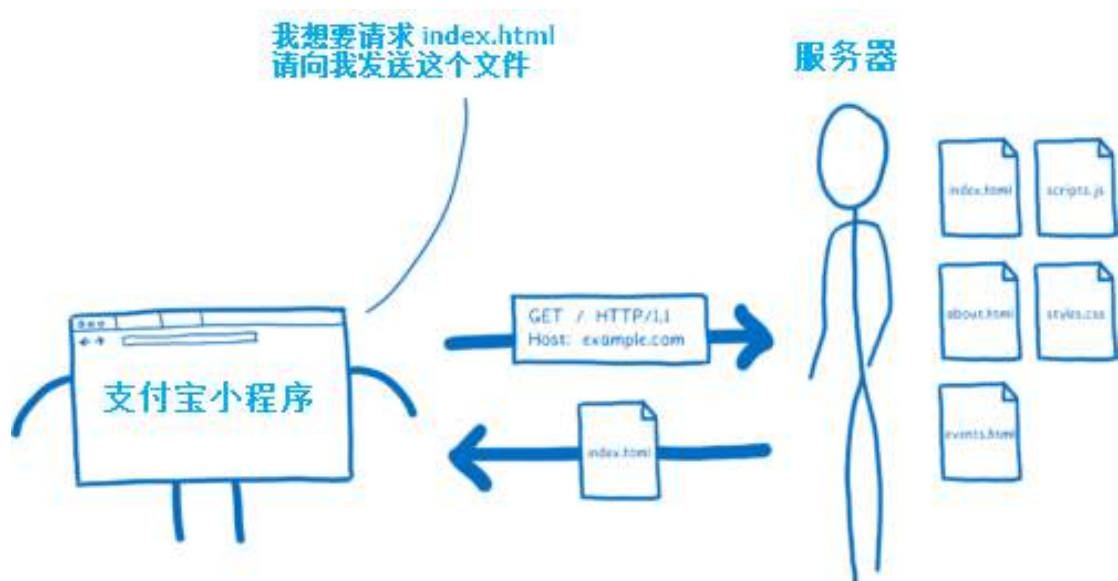
木心怀念着过去那个通过邮件传递信息的简单时代。

当下的网络时代虽然瞬息万变，但传递信息也是一样的“说一句，是一句”，“你锁了，人家就懂了”。

小程序经常需要往服务器传递数据或者从服务器拉取信息。

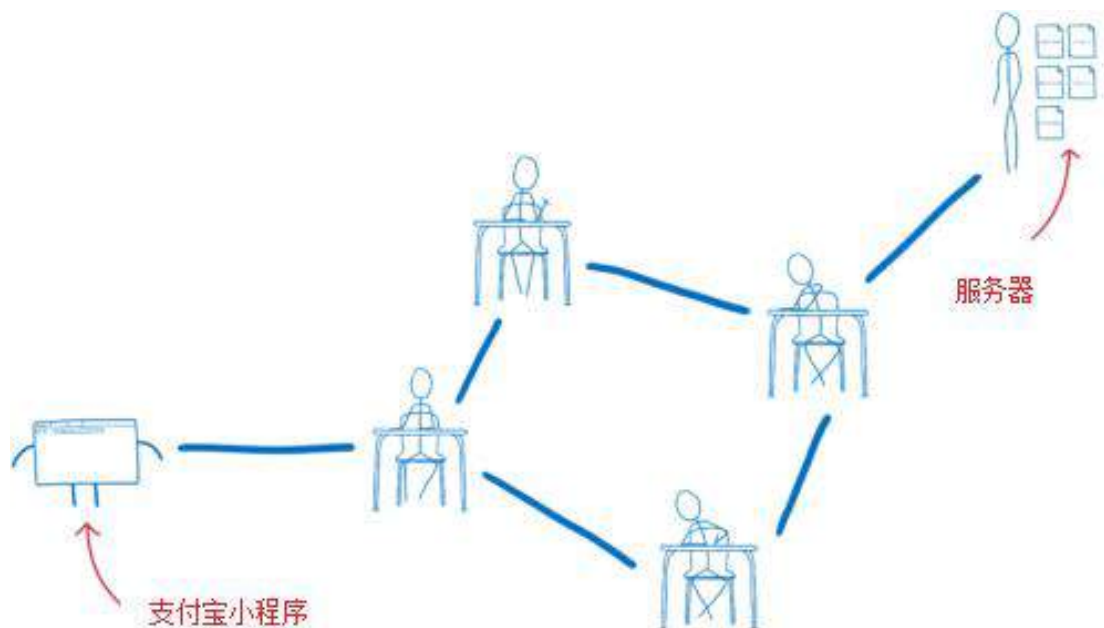
当用户通过小程序加载服务器传来的信息时，整个网络过程如下：

1. 用户通过小程序向服务器发出 GET 请求，
2. 服务器发送一个响应，响应信息包含一个数据文件。



该过程叫做 HTTP。

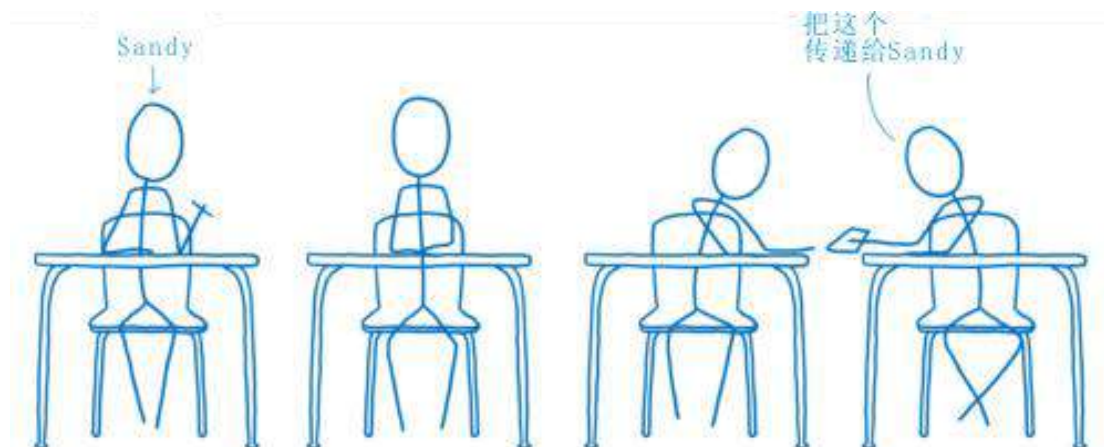
上面的流程也许过于简化，其实用户与服务器之间不可能面对面直接通话，因为它们相隔不是很近，甚至服务器是在浏览器的千里之外，而客户端浏览器不可能直通服务器。



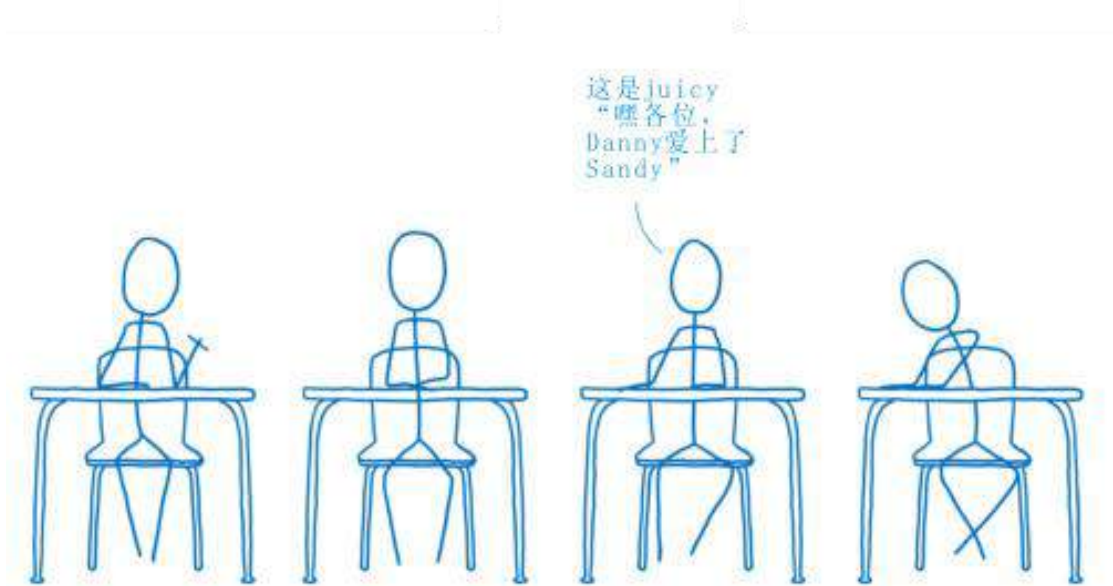
每一次的网络请求，小程序传递给服务器的信息，中间经过多重的信息转达。同理服务器回应小程序的响应也是同样的路径。

通俗点说，就是传纸条的原理。写字条的同学需要把字条递给旁边第一个同学，然后第二个同学递给第三个同学，以此类推，一直传递到最后的的信息接收者。

让我们看看，在传递字条的过程中，如果信息发出者想要给信息最末尾的接受者告白，会发生什么呢？



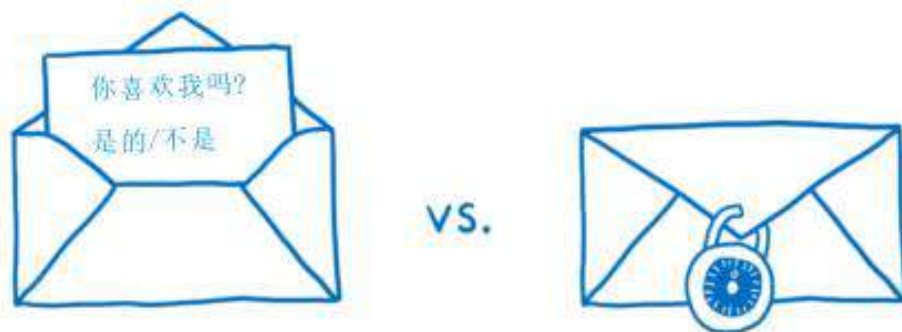
在 HTTP 状态下，传递者都可以打开字条，查看里面的内容。而且发送信息者无法知道传输路径，一旦发生信息窃取，甚至不知道是谁窃取的。



还是就是当信息落入心怀不轨的人手中，或者篡改信息内容，其后果不可设想。比如，把“你喜欢我吗？”篡改成“你不喜欢我吗？”。



为了避免这些情况发生，HTTP 安全版本应运而生，即 HTTPS。通过 HTTPS，传送的每次信息都被加上一个锁。



该锁配套的公钥和密钥仅小程序和服务端知道，其他传递者无法获取。因此，无论客户端发送的信息经过多个路由器，他人都无法读取信息内容。

客户端发送初始信息到服务器时，在信息内容中包含服务器的名称（在名为“服务器名称指示”的字段中）。而服务器运行商可以在同一台计算机上运行多个站点，



因此运行商可以跟踪到客户端的访问轨迹。虽然初始的信息已设置了加密，但是初始请求是仍未加密的。

这就是通过小程序 `my.request` 安全传递告白信息的故事。本节将介绍如何使用 `my.request` API 实现网络请求，并介绍一些使用注意事项。

**版本要求：**基础库 1.11.0 或更高版本；支付宝客户端 10.1.32 或更高版本，若版本较低，建议做 [兼容处理](#)。

发起网络请求：

- `my.request` 目前支持 GET/POST。
- `my.request` 目前只支持 HTTPS 协议的请求。

#### 使用说明：

- 请预先在 [支付宝小程序管理中心](#) > 小程序详情 > 设置 > 开发设置 > 服务器域名白名单中配置域名白名单。小程序在以下 API 调用时只能与白名单中的域名进行通讯：HTTP 请求（`my.request`）、上传文件（`my.uploadFile`）、下载文件（`my.downloadFile`）和 WebSocket（`my.connectSocket`）。



- 添加服务器域名白名单后，需要重新打包上传生成体验版，服务器域名才会生效。
- 在 IDE 上进行调试时，请使用真机预览调试。
- 支付宝客户端已不再维护 [my.httpRequest](#)，建议使用 `my.request`。另外，钉钉客户端尚不支持 `my.request`。若在钉钉客户端开发小程序，则需要使用 `my.httpRequest`。

## 扫码体验



#### 重要：

- 小程序开发过程中，可在开发工具内 **详情 > 域名信息 > 忽略 httpRequest 域名合法性检查** 中选择是否忽略域名合法性检查，如果选择忽略，则在模拟器、预览以及真机调试场景不会校验域名合法性，但小程序上线前必须确保通讯域名在白名单内，否则在正式版本无法调用。
- my.request 的请求头默认值为 {'content-type': 'application/json'}，而不是{'content-type': 'application/x-www-form-urlencoded'}。此外，请求头对象里面的 key 和 value 必须是 String 类型。

## 示例代码

```
// dataType 为 json 示例
my.request({
  url: 'https://httpbin.org/post',
  method: 'POST',
  data: {
    from: '支付宝',
    production: 'AlipayJSAPI',
  },
  dataType: 'json',
  success: function(res) {
    my.alert({content: 'success'});
  },
  fail: function(res) {
    my.alert({content: 'fail'});
  },
  complete: function(res) {
    my.hideLoading();
    my.alert({content: 'complete'});
  }
});

// dataType 为 base64 示例
my.request({
  url:
    'https://gw.alipayobjects.com/mdn/miniapp_de/afts/img/A*G1kWSJbe2zEAAAAA
    AAAAAABjARQnAQ',
  method: 'GET',
  dataType: 'base64',
```

```

success: (resp) => {
  console.log('resp data length', resp.data.length);
  console.log('resp data', resp.data); // 返回格式类似于:
data:image/png;base64,iVBORw0KG...
},
fail: (err) => {
  console.log('error', err);
},
});

```

## 入参

Object 类型，属性如下：

| 属性       | 类型       | 必填 | 描述   |
|----------|----------|----|--|
| url      | String   | 是  | 目标服务器 URL。   |
| headers  | Object   | 否  | 设置请求的 HTTP 头对象，默认 {'content-type': 'application/json'}，该对象里面的 key 和 value 必须是 String 类型。 |
| method   | String   | 否  | 默认 GET，目前支持 GET/POST/PUT/DELETE。   |
| data     | Object   | 否  | 详见 <b>data</b> 参数说明。   |
| timeout  | Number   | 否  | 超时时间，单位 ms，默认 30000。   |
| dataType | String   | 否  | 期望返回的数据格式，默认 JSON，支持 JSON、text、base64、arraybuffer（10.1.70 版本开始支持）。                       |
| success  | Function | 否  | 调用成功的回调函数。   |
| fail     | Function | 否  | 调用失败的回调函数。   |
| complete | Function | 否  | 调用结束的回调函数（调用成功、失败都会执行）。  |

## data 参数说明

传给服务器的数据最终会是 String 类型，如果 data 不是 String 类型，会被转换成 String。转换规则如下：

- 若方法为 GET，会将数据转换成 query string：  
encodeURIComponent(k)=encodeURIComponent(v)&encodeURIComponent(k)=encodeURIComponent(v)...
- 若方法为 POST 且 headers['content-type'] 为 application/json，会对数据进行 JSON 序列化
- 若方法为 POST 且 headers['content-type'] 为 application/x-www-form-urlencoded，会将数据转换成 query string：  
encodeURIComponent(k)=encodeURIComponent(v)&encodeURIComponent(k)=encodeURIComponent(v)...

## success 回调函数

入参为 Object 类型，属性如下：

| 属性      | 类型     | 描述  |
|---------|--------|---|
| data    | String | 响应数据，格式取决于请求时的 dataType 参数，<br>如果 dataType 值为 base64 时，返回的是符合 <a href="#">data URI scheme</a> 规范的内容字符串。 |
| status  | Number | 响应码。  |
| headers | Object | 响应头。  |

## 返回值 RequestTask

网络请求任务对象。调用 my.request 后返回的请求对象。

### RequestTask.abort()

中断请求任务。

### 示例代码

```
// 返回 RequestTask，可以调用 abort 方法取消请求
const task = my.request({url: 'https://httpbin.org/post'})
task.abort();
```

# “抱歉，不是我的菜”：小程序扫码点餐化解尴尬

月上柳梢头，人约黄昏后。

周五到了，小心翼翼约她出来吃晚饭，她欣然应约。

餐厅位于徐汇区闹中取静的华山路，法式梧桐的点缀让餐厅更显典雅，也更富有异国情调。踏入餐厅，灯光是橘色的，餐具是蓝的，桌椅也是蓝的，让人恍惚之间有了爱琴海边的错觉，唯美的装修风格、充满欧洲风味的精致美食，处处洋溢着地中海风情，真浪漫啊。

她翩翩而至，裙裾飞扬。

见到她我脸红了。我紧张地问她要吃些什么，又手忙脚乱地叫来服务员点完了菜，脸上冒出了小汗珠。

窗外的小雨滴滴答答，窗内的我们显得格外安静。

我鼓起勇气，打破沉默，小声问道：“你……你对我印象如何？”

“抱歉，不是我的菜……”

此刻，我如同五雷轰顶，只觉天旋地转，眼前华光溢彩的餐厅瞬间变得黯淡了。

“你是不是点错菜了？还是上错菜了呢？”她指着桌上的法式田螺和奶油蘑菇汤，瞪大了眼睛问我。旁边站着满脸疑惑的上菜员。

如何化解点错菜的尴尬呢？这时就需要使用支付宝小程序扫码点餐的功能了。

嘿！竟然可以扫码点菜了，我要试试！

扫码点餐



就要这个套餐好了...



顾客下单



后厨打单



顾客就餐



手机支付



完成消费

为了让用户减少输入，我们可以把复杂的信息编码成一个二维码，利用 `my.scan` API 调起支付宝扫一扫，用户扫码之后，`my.scan` 的 `success` 回调会收到这个二维码所对应的字符串信息。

例如餐厅点餐的小程序，我们给餐厅中每个餐桌编号 1-100 号，把这个数字编码到二维码中，扫码获得编号之后，就可以知道是哪一桌点的菜，大大提高点餐体验和效率。

```
// 利用 my.scanCode 获取二维码的数据
//page.js
Page({
  // 点击“扫码订餐”的按钮，触发 tapScan 回调
  tapScan: function() {
    // 调用 my.login 获取微信登录凭证
    my.scanCode({
      success: function(res) {
        var num = res.result // 获取到的 num 就是餐桌的编号
      }
    })
  }
})
```

还有很多场景可以结合支付宝 App 扫码能力做到很好的体验，例如通过扫商品上的一维码做一个商品展示的小程序；通过扫共享单车上的二维码去开启单车。我们可以多思考如何利用这个扫码能力去替代一些繁琐的输入操作，让我们的小程序变得更加便捷。

## 扫码体验



用支付宝扫一扫，进入小程序

## 示例代码

```
// API-DEMO page/API/scan-code/scan-code.json
{
  "defaultTitle": "Scan"
```

```
}
```

```
<!-- API-DEMO page/API/scan-code/scan-code.xml-->
<view class="page">
  <view class="page-section">
    <form onSubmit="scanCode">
      <view>
        <button type="primary" onTap="scan">扫码</button>
      </view>
    </form>
  </view>
</view>
```

```
// API-DEMO page/API/scan-code/scan-code.js
Page({
  scan() {
    my.scan({
      type: 'qr',
      success: (res) => {
        my.alert({ title: res.code });
      },
    });
  }
})
```

## 入参

Object 类型，属性如下：

| 属性        | 类型       | 必填 | 描述   |
|-----------|----------|----|--|
| type      | String   | 否  | 扫码样式（默认 qr）：<br>qr：扫码框样式为二维码扫码框。<br>bar：扫码样式为条形码扫码框。 |
| hideAlbum | Boolean  | 否  | 是否隐藏相册（不允许从相册选择图片），只能从相机扫码。                          |
| success   | Function | 否  | 调用成功的回调函数。   |



|          |          |   |                         |
|----------|----------|---|-------------------------|
| fail     | Function | 否 | 调用失败的回调函数。              |
| complete | Function | 否 | 调用结束的回调函数（调用成功、失败都会执行）。 |

## success 回调函数

入参为 Object 类型，属性如下：

| 属性      | 类型     | 描述             |
|---------|--------|----------------|
| code    | String | 扫码所得数据。        |
| qrCode  | String | 扫描二维码时返回二维码数据。 |
| barCode | String | 扫描条形码时返回条形码数据。 |

# 定格甜蜜回忆：小程序上传图片功能实现

还记得你们第一次手牵手出门吗？

还记得你们第一次共享浪漫的烛光晚餐吗？

还记得你们第一次依偎在电影院中约会吗？

还记得你们第一次去迪士尼约会圆她一个公主梦吗？

甜蜜的回忆，多想定格在时光里。不如开发一个小程序，用于上传存储这些甜甜的照片吧！

本节将介绍如何通过 my.uploadFile API 实现小程序上传图片的功能。

**使用前提：**

请预先在 [支付宝小程序管理中心](#) > 小程序详情 > 设置 > 开发设置 > 服务器域名白名单 中配置域名白名单。小程序在以下 API 调用时只能与白名单中的域名进行通讯：HTTP 请求（my.request）、上传文件（my.uploadFile）、下载文件（my.downloadFile）和 WebSocket（my.connectSocket）。



## 扫码体验



## 示例代码

```
// API-DEMO page/upload-file/upload-file.json
{
  "defaultTitle": "Upload File"
}
```

```
<!-- API-DEMO page/upload-file/upload-file.xml -->
```

```
<view class="page">
  <button type="primary" onTap="uploadFile">上传图片</button>
</view>
```

```
// API-DEMO page/API/upload-file/upload-file.js
```

```
Page({
  uploadFile() {
    my.chooseImage({
      chooseImage: 1,
      success: res => {
        const path = res.apFilePaths[0];
        console.log(path);
        my.uploadFile({
          url: 'http://httpbin.org/post',
          fileType: 'image',
          fileName: 'file',
          filePath: path,
          success: res => {
            my.alert({ title: '上传成功' });
          },
          fail: function(res) {
            my.alert({ title: '上传失败' });
          },
        });
      },
    });
  },
});
```

上传文件的后端代码，相关 openAPI 请参考

[alipay.offline.material.image.upload\(上传门店照片和视频接口\)](#)。

```
@Override
```

```
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
```

```
    String path = req.getParameter("filePath");
```

```
    //得到要下载的文件名
```

```
    String fileName = URLEncoder.encode(req.getParameter("fileName"), "utf-8");
```

```

String fileType = path.substring(path.lastIndexOf('.')+1,path.length());
FileInputStream fis = new FileInputStream(path);

System.out.println("debugFileName: "+ fileName);
// 下载文件存放路径
String localPath = "";

FileOutputStream fs = new FileOutputStream(localPath + fileName
+ "." + fileType);

resp.setHeader("content-disposition", "attachment;filename="+fileName);
resp.setHeader("content-type", fileType );

// 执行 fileOutputStream 的输出操作
int len = 1;
byte[] b = new byte[1024];
while((len=fis.read(b))!=-1){
    fs.write(b, 0, len);
}
fs.close();
fis.close();
}

```

## 入参

Object 类型，属性如下：

| 属性       | 类型     | 必填 | 描述  |
|----------|--------|----|---|
| url      | String | 是  | 开发者服务器地址。                                   |
| filePath | String | 是  | 要上传文件资源的本地定位符。                              |
| fileName | String | 是  | 文件名，即对应的 key，开发者在服务器端通过这个 key 可以获取到文件二进制内容。 |
| fileType | String | 是  | 文件类型支持图片、视频、音频（ image / video / audio ）     |

|          |          |   |                         |
|----------|----------|---|-------------------------|
| header   | Object   | 否 | HTTP 请求 Header。         |
| formData | Object   | 否 | HTTP 请求中其他额外的 form 数据。  |
| success  | Function | 否 | 调用成功的回调函数。              |
| fail     | Function | 否 | 调用失败的回调函数。              |
| complete | Function | 否 | 调用结束的回调函数（调用成功、失败都会执行）。 |

## success 回调函数

入参为 Object 类型，属性如下：

| 属性         | 类型     | 描述             |
|------------|--------|----------------|
| data       | String | 服务器返回的数据。      |
| statusCode | String | HTTP 状态码。      |
| header     | Object | 服务器返回的 header。 |

# 爱情不掉线：小程序获取设备的网络状态

地球不爆炸，爱情不掉线。宇宙不重启，我们不分离。

不想要和手机里的女朋友掉线的你，如何获取到她的手机网络状态呢？



大家都知道，手机连接到互联网有几种方式：WiFi、2G、3G、4G，包括很快到来的 5G。每种方式的上传速度和下载速度差异很大，它们的计费方式的差异也导致用户在使用互联网服务的时候有不同的使用习惯。

WiFi 相对于其他几种网络连接方式，其速度会更快。WiFi 一般情况下，都是免费供用户使用，而移动数据网络是需要根据使用流量进行计费的。

考虑这样一个场景，小程序需要下载一些文档，然后通过小程序的能力去预览这个文档，这些文档可能文件体积比较大，对于某些用户来说，他们并不想耗费太多的数据流量去预览文档。这种情况下，可以通过小程序提供的获取网络状态 API `my.getNetworkType`，做一些更友好的体验提示。

## 扫码体验



用支付宝扫一扫，进入小程序

## 入参

Object 类型，属性如下：

| 属性       | 类型       | 必填 | 描述                      |
|----------|----------|----|-------------------------|
| success  | Function | 否  | 调用成功的回调函数。              |
| fail     | Function | 否  | 调用失败的回调函数。              |
| complete | Function | 否  | 调用结束的回调函数（调用成功、失败都会执行）。 |

## success 回调函数

入参为 Object 类型，属性如下：

| 属性               | 类型      | 描述  |
|------------------|---------|---|
| networkAvailable | Boolean | 网络是否可用。   |
| networkType      | String  | 网络类型值 UNKNOWN / NOTREACHABLE / WIFI / 3G / 2G / 4G / WWAN |

## 示例代码

```
// API-DEMO page/API/get-network-type/get-network-type.json
{
  "defaultTitle": "获取手机网络状态"
}
```

```
<!-- API-DEMO page/API/get-network-type/get-network-type.xml-->
<view class="page">
  <view class="page-section">
    <view class="page-section-demo">
      <view class="page-body-title">网络状态</view>
      <block a:if="{{hasNetworkType === false}}">
        <text class="page-body-text">未获取</text>
        <text class="page-body-text">点击按钮可获取网络状态</text>
      </block>
      <block a:if="{{hasNetworkType === true}}">
        <text class="page-body-text-network-type">{{networkType}}</text>
      </block>
    </view>
  </view>
</view>
```

```

    </block>
  </view>
  <view class="page-section-btns">
    <view onTap="getNetworkType">获取手机网络状态</view>
    <view onTap="clear">清空</view>
  </view>
</view>
</view>

```

```

// API-DEMO page/API/get-network-type/get-network-type.js
Page({
  data: {
    hasNetworkType: false
  },
  onLoad() {
    this.onChange = this.onChange.bind(this);
    // my.onNetworkChange(this.onChange);
  },
  onChange(res){
    console.log('onNetworkChange', res);
    this.setData({
      hasNetworkType: true,
      networkType: res.networkType
    });
  },
  onUnload() {
    // my.offNetworkChange(this.onChange);
  },
  getNetworkType() {
    my.getNetworkType({
      success: (res) => {
        this.setData({
          hasNetworkType: true,
          networkType: res.networkType
        })
      }
    })
  }
})

```



```
},  
clear() {  
  this.setData({  
    hasNetworkType: false,  
    networkType: ''  
  })  
},  
});
```

```
/* API-DEMO page/API/get-network-type/get-network-type.acss */  
.page-body-info {  
  height: 200px;  
}  
.page-body-text-network-type {  
  font-size: 80px;  
  font-family: Helvetica;  
}
```

# 支付宝小程序组件

## 初次见面

只因为在蚂蚁群中多看了你一眼，就难以忘掉你容颜.....小红小蓝的相遇就是这么的狗血，就是蚁海中的擦肩而过回眸那一眼，小蓝大脑弹出提示：这个蚂蚁美眉真是蚁群中最亮眼的那一只呀。小红心里想：这只蚂蚁哥哥真是独特呀。至于这劫能不能逃得过，请见下回分解。下面我们讲一下关于 tips 提示的相关信息。

Tips 引导是特定应用场景下系统针对用户的一种功能引导方式，例如用户第一次登录后、或者某个新功能上线后的提示等。分为两种类型：tipsdialog（对话型）、tipsplain（简单型）。

## 扫码体验



## 示例代码

```
{
  "defaultTitle": "Tips",
  "usingComponents": {
    "tips-dialog": "mini-ali-ui/es/tips/tips-dialog/index",
    "tips-plain": "mini-ali-ui/es/tips/tips-plain/index"
  }
}
```

### tips-dialog

```
<view>
  <tips-dialog
    show="{{showDialog}}"
    className="dialog"
    type="dialog"
```

```

>
<view class="content" slot="content">
  <view>hello,</view>
  <view>欢迎使用小程序扩展组件库 mini-ali-ui</view>
</view>
<view slot="operation" class="opt-button" onTap="onDialogTap">知道了</view>
</tips-dialog>
<tips-dialog
  imageUrl="https://gw.alipayobjects.com/zos/rmsportal/AzRAgQXlnNbEwQRvEwiu.png"
  type="rectangle"
  className="rectangle"
  onCloseTap="onCloseTap"
  show="{{showRectangle}}">
  <view class="content" slot="content">
    把“城市服务”添加到首页
  </view>
  <view slot="operation" class="add-home" onTap="onRectangleTap">立即添加
</view>
</tips-dialog>
</view>

```

```

Page({
  data: {
    showRectangle: true,
    showDialog: true,
  },
  onCloseTap() {
    this.setData({
      showRectangle: false,
    });
  },
  onRectangleTap() {
    my.alert({
      content: 'do something',
    });
  }
});

```

```
},  
onDialogTap() {  
  this.setData({  
    showDialog: false,  
  });  
},  
});
```

```
.rectangle {  
  position: fixed;  
  bottom: 100px;  
}  
.dialog {  
  position: fixed;  
  bottom: 10px;  
}  
.content {  
  font-size: 14px;  
  color: #fff;  
}  
.opt-button {  
  width: 51px;  
  height: 27px;  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  color: #fff;  
  font-size: 12px;  
  border: #68BAF7 solid 1rpx;  
}  
.add-home {  
  width: 72px;  
  height: 27px;  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  background-color: #56ADEB;
```

```
color: #fff;
font-size: 14px;
}
```

## tips-plain

```
<tips-plain onClose="onClose" time="{{time}}">{{content}}</tips-plain>
Page({
  data: {
    content: '我知道了',
    time: 2000,
  },
  onClose() {
    my.alert({
      title: '12321'
    });
  }
});
```

## 属性

### tips-dialog

| 属性         | 类型        | 默认值    | 说明                                     | 必填 |
|------------|-----------|--------|--|----|
| className  | String    | -      | 自定义 class。                             | 否  |
| show       | Boolean   | true   | 控制组件是否展示。                              | 否  |
| type       | String    | dialog | dialog 表示对话框的样式类型，rectangle 表示矩形的样式类型。 | 否  |
| onCloseTap | ()=> void | -      | 当 type 值为 rectangle 时，组件点击关闭 icon 的回调。 | 否  |
| iconUrl    | String    | -      | 展示 icon 的 url 地址。                      | 否  |

|               |        |             |   |   |
|---------------|--------|-------------|---|---|
| arrowPosition | String | bottom-left | 控制 tips 中的箭头位置。可选值：bottom-left、bottom-center、bottom-right、top-left、top-center、top-right、left、right。 | 否 |
|---------------|--------|-------------|---|---|

## slots

|           |             |
|-----------|-------------|
| slotName  | 说明          |
| content   | 用于渲染提示的正文内容 |
| operation | 用于渲染右侧操作区域  |

## tips-plain

| 属性        | 类型         | 默认值   | 描述             | 默认值      |
|-----------|------------|-------|----------------|----------|
| className | String     | false | 自定义 class。     | -        |
| time      | Number     | false | 自动关闭时间（单位：毫秒）。 | 5000(ms) |
| onClose   | () => void | false | 回调并关闭提示框。      | -        |

## 对方向你发出了好友邀请

作为男士，小蓝主动发起了对话：小红美眉，我可以加你为支付宝好友吗？小红：用来作什么呢？小蓝：我可以天天送你能量跟鸡饲料，种树献爱心。小红心想自己想种一棵樟子松还差点能量，加就加。至此，小红小蓝成功成为了支付宝好友。

当应用中需要比较明显的对用户当前的操作行为进行警示或提醒时，可以使用对话框。用户需要针对对话框进行操作后方可结束。

## 扫码体验



用支付宝扫一扫，进入小程序

## 示例代码

```
{
  "defaultTitle": "Modal",
  "usingComponents": {
    "modal": "mini-ali-ui/dist/es/modal/index"
  }
}
```

```
<view>
  <button onTap="openModal">打开 modal</button>
  <modal
    show="{{modalOpened}}"
    onModalClick="onModalClick"
    onModalClose="onModalClose"

    topImage="https://gw.alipayobjects.com/zos/rmsportal/yFeFExbGpDxvDYnKHcrs.
    png"
  >
    <view slot="header">标题单行</view>
    说明当前状态、提示用户解决方案，最好不要超过两行。
    <view slot="footer">确定</view>
```

```
</modal>
</view>
```

```
Page({
  data: {
    modalOpened: false,
  },
  openModal() {
    this.setData({
      modalOpened: true,
    });
  },
  onModalClick() {
    this.setData({
      modalOpened: false,
    });
  },
  onModalClose() {
    this.setData({
      modalOpened: false,
    });
  }
});
```

## 属性

| 属性名       | 类型      | 默认值   | 描述                             |
|-----------|---------|-------|--------------------------------|
| className | String  | -     | 自定义 class。                     |
| show      | Boolean | false | 是否展示 modal。<br>可选值：true、false。 |
| showClose | Boolean | true  | 是否渲染 关闭。<br>可选值：true、false。    |
| mask      | Boolean | true  | 是否展示蒙层。<br>可选值：true、false。     |
| closeType | String  | 0     | 关闭图表类型。                        |



|               |               |                     |  |
|---------------|---------------|---------------------|--|
|               |               |                     | 可选值：0-灰色图标；1-白色图标。   |
| onModalClick  | EventHandle   | () => void          | 选择区间时的回调。  |
| onModalClose  | EventHandle   | () => void          | 点击 <b>关闭</b> 的回调。showClose 为 false 时无需设置。                      |
| topImage      | String        | -                   | 顶部图片。  |
| topImageSize  | String        | md                  | 顶部图片规则。<br>可选值：<br>lg (带图弹框-大图);<br>md (带图弹框);<br>sm(带图弹框-小图)。 |
| buttons       | Array<Object> | md                  | 底部自定义多按钮, 详情见 buttons 配置。                                      |
| onButtonClick | EventHandle   | (e: Object) => void | 点击 buttons 部分的回调。  |
| buttonsLayout | String        | horizontal          | 设置 buttons 的对齐方式。<br>可选值：horizontal, vertical。                 |
| advice        | Boolean       | false               | 是否是运营类弹窗。<br>可选值：true、false。                                   |
| zIndex        | String/Number | -                   | 设置弹框层级。  |
| disableScroll | Boolean       | false               | modal 展示时是否禁止页面滚动（以真机效果为准）。<br>可选值：true、false。                 |

## buttons

提供按钮组配置，每一项表示一个按钮。

| 属性名 | 类型 | 描述 |
|-----|----|----|
|-----|----|----|

|          |        |                         |
|----------|--------|-------------------------|
| text     | String | 按钮的文本。                  |
| extClass | String | 按钮自定义 Class, 可用户定制按钮样式。 |

## slots

| slotName | 描述        | 必填    |
|----------|-----------|-------|
| header   | modal 头部。 | false |
| footer   | modal 尾部。 | false |

## 开启了愉快的聊天模式

看到小红通过了它的好友申请，小蓝赶紧发起聊天：小红美眉，你平时喜欢做什么呀，有空我们可以一起约呀。此处就可以使用 input 输入框啦，不仅可以输入文字还可以输入数字、密码哦。

输入框，可设置输入内容的类型、长度、显示形式等。当用户需要输入文字内容时点击文本框，它将自动打开键盘。使用文本字段来请求少量信息。

注意：

- iOS 系统支付宝客户端版本 10.1.80 及以上不支持 focus=true 自动唤起。
- 小程序中 input 如果父类是 position: fixed，可以加上 enableNative="{{false}}", 解决输入框错位问题。

## 扫码体验



用支付宝扫一扫，进入小程序

## 示例代码

```
<!-- API-DEMO page/component/input/input.xml -->
<view class="page">
  <view class="page-description">输入框</view>
  <view class="page-section">
    <view class="form-row">
      <view class="form-row-label">受控聚焦</view>
      <view class="form-row-content">
        <input class="input" focus="{{focus}}" onFocus="onFocus" onBlur="onBlur"
placeholder="input something" />
      </view>
    </view>
  </view>
  <view class="page-section-btns">
    <button size="mini" onTap="bindButtonTap">聚焦</button>
  </view>
```

```

</view>
<view class="page-section">
  <view class="form-row">
    <view class="form-row-label"><label for="controlled">显示输入
</label></view>
    <view class="form-row-content">
      <input class="input" id="controlled" onInput="bindKeyInput"
placeholder="show input content" />
    </view>
  </view>
  <view class="extra-info">你输入的是: {{inputValue}}</view>
</view>
<view class="page-section">
  <view class="form-row">
    <view class="form-row-label">最大长度</view>
    <view class="form-row-content">
      <input class="input" maxlength="10" placeholder="maxlength 10" />
    </view>
  </view>
  <view class="form-line" />
  <view class="form-row">
    <view class="form-row-label">收起键盘</view>
    <view class="form-row-content">
      <input class="input" onInput="bindHideKeyboard" placeholder="输入 123
自动收起键盘" />
    </view>
  </view>
  <view class="form-line" />
  <view class="form-row">
    <view class="form-row-label">输入密码</view>
    <view class="form-row-content">
      <input class="input" password type="text" placeholder="密码输入框" />
    </view>
  </view>
  <view class="form-line" />
  <view class="form-row">
    <view class="form-row-label">输入数字</view>

```

```

    <view class="form-row-content">
      <input class="input" type="number" placeholder="数字输入框" />
    </view>
  </view>
  <view class="form-line" />
  <view class="form-row">
    <view class="form-row-label">小数点键盘</view>
    <view class="form-row-content">
      <input class="input" type="digit" placeholder="带小数点的数字键盘" />
    </view>
  </view>
  <view class="form-line" />
  <view class="form-row">
    <view class="form-row-label">身份证键盘</view>
    <view class="form-row-content">
      <input class="input" type="idcard" placeholder="身份证输入键盘" />
    </view>
  </view>
</view>
<view class="page-section">
  <view class="page-section-title">搜索框</view>
  <view class="page-section-demo">
    <view class="search-outer">
      <input
        class="search-input"
        placeholder="搜索"
        value="{{search}}"
        onConfirm="doneSearch"
        onInput="handleSearch"
      />
      <text class="search-cancel" onTap="clearSearch">取消</text>
    </view>
  </view>
</view>
</view>

```

```
// API-DEMO page/component/input/input.js
```

```

Page({
  data: {
    focus: false,
    inputValue: '',
  },
  bindButtonTap() {
    // blur 事件和这个冲突
    setTimeout(() => {
      this.onFocus();
    }, 100);
  },
  onFocus() {
    this.setData({
      focus: true,
    });
  },
  onBlur() {
    this.setData({
      focus: false,
    });
  },
  bindKeyInput(e) {
    this.setData({
      inputValue: e.detail.value,
    });
  },
  bindHideKeyboard(e) {
    if (e.detail.value === '123') {
      // 收起键盘
      my.hideKeyboard();
    }
  },
  handleSearch(e) {
    console.log('search', e.detail.value);
    this.setData({
      search: e.detail.value,
    });
  }
});

```

```

    },
    doneSearch() {
      console.log('doneSearch', this.data.search);
      my.hideKeyboard();
    },
    clearSearch() {
      console.log('clear search', this.data.search);
      this.setData({
        search: '',
      });
    },
  },
});

```

```

/* API-DEMO page/component/input/input.acss */
.extra-info {
  border-top: 1px solid #ddd;
  margin-left: 30px;
  padding: 20px 0;
  overflow: auto;
}
.search-outer {
  box-sizing: border-box;
  display: flex;
  height: 40px;
  overflow: hidden;
  padding: 8px;
  border-bottom: 1px solid #ddd;
  background-color: #eff4;
}
.search-outer * {
  box-sizing: border-box;
}
.search-input {
  flex: 1;
  text-align: left;
  display: block;
  color: #000;

```

```

height: 24px;
font-size: 15px;
background-color: #fff;
border-color: transparent;
}
.search-input:focus + .search-cancel {
margin-right: 0;
opacity: 1;
}
.search-cancel {
margin-right: -40px;
display: inline-block;
opacity: 0;
padding-left: 8px;
height: 24px;
line-height: 24px;
font-size: 16px;
color: #108ee9;
text-align: right;
transition: margin-right .3s, opacity .3s;
transition-delay: .1s;
}

```

## 属性

| 属性    | 类型     | 默认值  | 描述   | 最低版本  |
|-------|--------|------|--|---|
| value | String | -    | 初始内容   | -   |
| name  | String | -    | 组件名字，用于表单提交获取数据  | -   |
| type  | String | text | input 的类型，有效值：text、number、idcard、digit(可以唤起带有小数点的数字键盘)、numberpad、digitpad、idcardpad。 | numberpad、digitpad、idcardpad 基础库 <a href="#">1.13.0</a> 客户端 |



|                   |         |       |  |  |
|-------------------|---------|-------|--|--|
|                   |         |       |  | 10.1.50, 可通过 <a href="#">my.canIUse(input.type.numberpad)</a> 来检测。 |
| password          | Boolean | false | 是否是密码类型  | -  |
| placeholder       | String  | -     | 占位符  | -  |
| placeholder-style | String  | -     | 指定 placeholder 的样式, 可设置间距  | <a href="#">1.6.0</a>  |
| placeholder-class | String  | -     | 指定 placeholder 的样式类  | <a href="#">1.6.0</a>  |
| disabled          | Boolean | false | 是否禁用   | -  |
| maxlength         | Number  | 140   | 最大长度   | -  |
| focus             | Boolean | false | 获取焦点   | -  |
| confirm-type      | String  | done  | 设置键盘右下角按钮的文字, 有效值: done (显示“完成”)、go (显示“前往”)、next (显示“下一个”)、search (显示“搜索”)、send (显示“发送”), 平台不同显示的文字略有差异。 <b>注意: 只有在 type=text 时有效</b> | <a href="#">1.7.0</a>  |
| confirm-hold      | Boolean | false | 点击键盘右下角按钮时是否保持键盘不收起状态  | <a href="#">1.7.0</a>  |
| cursor            | Number  | -     | 指定 focus 时的光标位置  | -  |
| selection-start   | Number  | -1    | 获取光标时, 选中文本对应的焦点光标起始位置, 需要和 selection-end 配合使用   | <a href="#">1.7.0</a>  |

|               |              |       |  |                       |
|---------------|--------------|-------|--|-----------------------|
| selection-end | Number       | -1    | 获取光标时，选中文本对应的焦点光标结束位置，需要和 selection-start 配合使用 | <a href="#">1.7.0</a> |
| randomNumber  | Boolean      | false | 当 type 为 number, digit, idcard 数字键盘是否随机排列      | <a href="#">1.9.0</a> |
| controlled    | Boolean      | false | 是否为受控组件。为 true 时，value 内容会完全受 setData 控制       | <a href="#">1.8.0</a> |
| onInput       | EventHandler | -     | 键盘输入时触发 input 事件，event.detail = {value: value} | -                     |
| onConfirm     | EventHandler | -     | 点击键盘完成时触发，event.detail = {value: value}        | -                     |
| onFocus       | EventHandler | -     | 聚焦时触发，event.detail = {value: value}            | -                     |
| onBlur        | EventHandler | -     | 失去焦点时触发（仅支持真机），event.detail = {value: value}   | -                     |

# 手拉手你是我的好朋友

时间在点击一个个发送按钮中不知不觉流逝，小蓝与小红的友谊不断加深，小小的按钮连接着他们彼此，我们来了解一下 button 按钮的使用方法。

需要重点强调该操作并且引导用户去点击的入口通过按钮表达。

## 扫码体验



用支付宝扫一扫，进入小程序

## 示例代码

```
<!-- API-DEMO page/component/button/button.xml -->
<view class="page">
  <view class="page-description">按钮</view>
  <view class="page-section">
    <view class="page-section-title">type-primary/ghost</view>
    <view class="page-section-demo">
      <button type="primary">主要操作 Normal</button>
      <button type="primary" loading>操作</button>
      <button type="primary" disabled>主要操作 Disable</button>
      <button type="ghost">ghost 操作</button>
      <button type="ghost" loading>ghost 操作</button>
      <button type="ghost" disabled>ghost 操作 Disable</button>
    </view>
  </view>
  <view class="page-section">
    <view class="page-section-title">type-default</view>
    <view class="page-section-demo">
      <button data-aspn-click="xxx">辅助操作 Normal</button>
      <button disabled>辅助操作 Disable</button>
    </view>
  </view>
  <view class="page-section">
```

```

<view class="page-section-title">type-warn</view>
<view class="page-section-demo">
  <button type="warn">警告类操作 Normal</button>
  <button type="warn" disabled>警告类操作 Disable</button>
  <button type="warn" hover-class="red">hover-red</button>
</view>
</view>
<view class="page-section">
  <view class="page-section-title">Size</view>
  <view class="page-section-demo">
    <button size="mini" loading>提交</button>
    <button style="margin-left: 10px;" type="primary" size="mini">选项</button>
  </view>
</view>
<view class="page-section">
  <view class="page-section-title">open</view>
  <view class="page-section-demo">
    <button open-type="share">share</button>
  </view>
</view>
<view class="page-section">
  <view class="page-section-title">Form</view>
  <view class="page-section-demo">
    <form onSubmit="onSubmit" onReset="onReset">
      <button form-type="submit" type="primary">submit</button>
      <button form-type="reset">reset</button>
    </form>
  </view>
</view>
</view>

```

```

// API-DEMO page/component/button/button.js
Page({
  data: {},
  onSubmit() {
    my.alert({ title: 'You click submit' });
  },

```

```
onReset() {
  my.alert({ title: 'You click reset' });
},
});
```

```
/* API-DEMO page/component/button/button.acss */
.red {
  background-color: red;
  border-color: red;
  color: #fff;
}

button + button {
  margin-top: 32rpx;
}
```

## 属性

| 属性          | 类型      | 默认值          | 描述   | 最低版本 |
|-------------|---------|--------------|--|------|
| size        | String  | default      | 有效值 default, mini (小尺寸)。   | -    |
| type        | String  | default      | 按钮的样式类型，有效值 primary, default,, warn。   | -    |
| plain       | Boolean | false        | 是否镂空   | -    |
| disabled    | Boolean | false        | 是否禁用   | -    |
| loading     | Boolean | false        | 按钮文字前是否带 loading 图标。   | -    |
| hover-class | String  | button-hover | 按钮按下去的样式类。button-hover 默认为 {background-color: rgba(0, 0, 0, 0.1); opacity: 0.7;}, hover-class="none" 时表示没有被点击效果。 | -    |

|                        |              |       |   |                        |
|------------------------|--------------|-------|---|------------------------|
| hover-start-time       | Number       | 20    | 按住后多少时间后出现点击状态，单位毫秒。  | -                      |
| hover-stay-time        | Number       | 70    | 手指松开后点击状态保留时间，单位毫秒。   | -                      |
| hover-stop-propagation | Boolean      | false | 是否阻止当前元素的祖先元素出现被点击样式。   | <a href="#">1.10.0</a> |
| form-type              | String       | -     | 有效值：submit, reset, 用于 <a href="#">form 表单</a> 组件，点击分别会触发 submit/reset 事件。 | -                      |
| open-type              | String       | -     | 开放能力  | <a href="#">1.1.0</a>  |
| scope                  | String       | -     | 当 open-type 为 getAuthorize 时有效。   | <a href="#">1.11.0</a> |
| onTap                  | EventHandler | -     | 点击  | -                      |
| app-parameter          | String       | -     | 打开 APP 时，向 APP 传递的参数，open-type="launchApp" 时有效。                           | -                      |
| public-id              | String       | -     | 生活号 id，必须是当前小程序同主体且已关联的生活号，open-type="lifestyle" 时有效。                     | -                      |

## open-type 有效值

| 值            | 说明  | 最低版本                   |
|--------------|---|------------------------|
| share        | 触发 <a href="#">自定义分享</a> ，可使用 <a href="#">my.canIUse</a> ('button.open-type.share') 判断  | <a href="#">1.1.0</a>  |
| getAuthorize | 支持小程序授权，可使用 <a href="#">my.canIUse</a> ('button.open-type.getAuthorize') 判断             | <a href="#">1.11.0</a> |
| contactShare | 分享到通讯录好友，可使用 <a href="#">my.canIUse</a> ('button.open-type.contactShare') 判断            | <a href="#">1.11.0</a> |
| lifestyle    | <a href="#">关注生活号</a> ，可使用 <a href="#">my.canIUse</a> ('button.open-type.lifestyle') 判断 | <a href="#">1.11.0</a> |

## scope 有效值

当 open-type 为 getAuthorize 时，可以设置 scope 为以下值：

| 值                           | 说明                            | 最低版本                   |
|-----------------------------|-------------------------------|------------------------|
| <a href="#">phoneNumber</a> | 唤起授权界面，用户可以授权小程序获取用户手机号。      | <a href="#">1.11.0</a> |
| <a href="#">userInfo</a>    | 唤起授权界面，用户可以授权小程序获取支付宝会员的基础信息。 | <a href="#">1.11.0</a> |

# 支付宝小程序开放能力

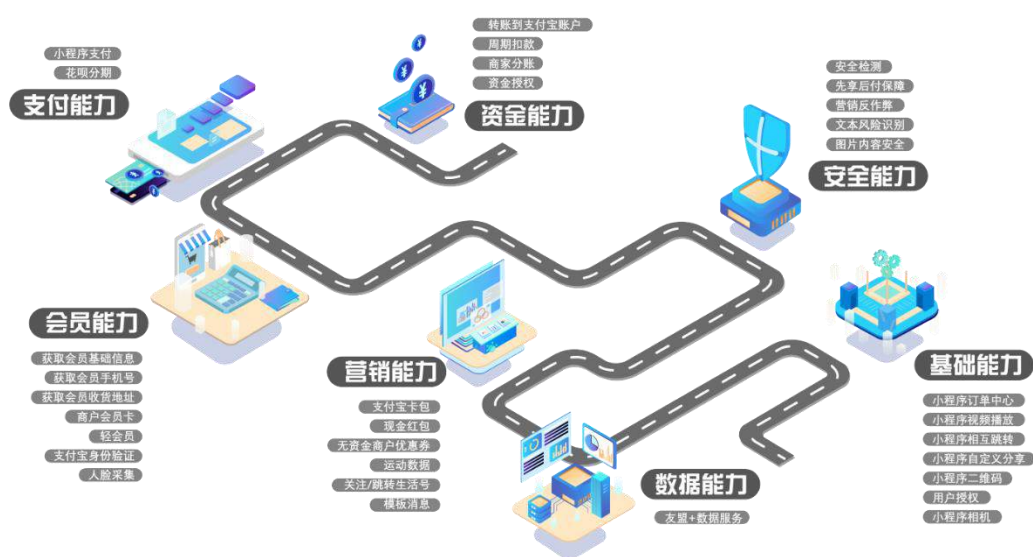
## 能力地图，让你的小程序拥有灵魂

开放能力，是支付宝开放平台为满足开发者不同的业务需求，将支付宝的一些功能以接口（API）的形式开放出来；开发者通过这些开放的接口与支付宝服务端（也有部分是支付宝客户端）进行交互，实现业务逻辑。

举个例子，假设你是老王，你开发的小程序是用来卖瓜的。那你要让你的用户在你的小程序内花钱买你的瓜，这样你就要接入“[小程序支付](#)”能力；你要在小程序内营销你的瓜，那么就录个带货的视频吧，那么就要使用“[小程序视频播放](#)”能力。总之，你需要贴合你的小程业务需求，不断地“武装”你的小程序。而开放能力也是你的“军火库”，为你提供源源不断的业务支持。

目前，小程序提供如下的开放能力，这些能力构成了一张“能力地图”。根据这张地图，你的小程序可以走出自己的路。

随着业务的不断发展，“能力地图”的版图会越来越大，为开发者提供更多福利。



点击进入 [能力中心](#) PC 端；或扫码收藏 [能力中心](#) 小程序，随时关注能力动态；分享能力更方便。



用支付宝扫一扫，进入小程序



# 能力实战

本章将重点介绍“获取会员基础信息”和“模板消息”的接入。

## 缘起：获取会员基础信息

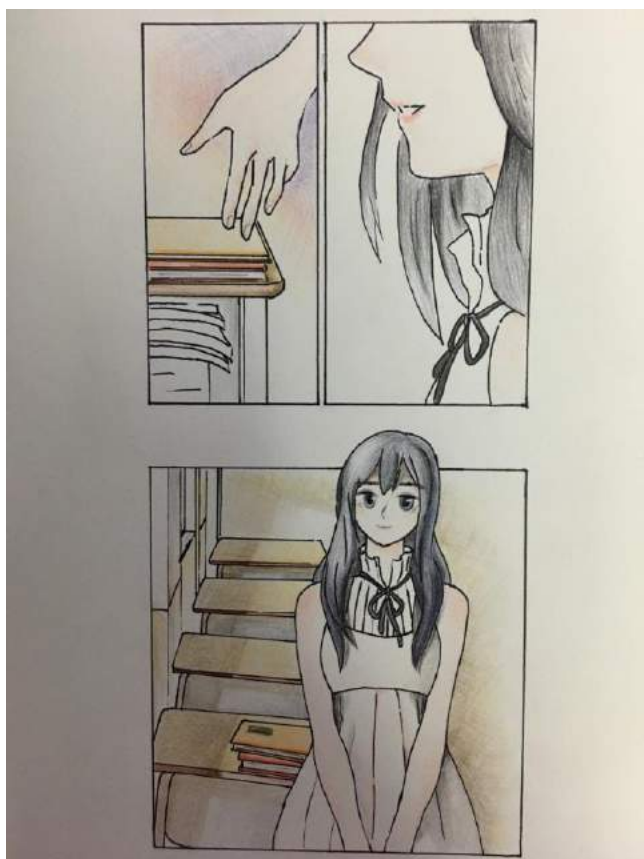
盛夏的图书馆总是学生们避暑的热门景点，对软件学院的大二学生小宝👦来说也不例外。窗外的知了没完没了的絮叨着，小宝双手赌气似的敲打着键盘，指尖生了魔法似的溜出一行行代码。刚写完一整段的小明将面前的冰咖啡一饮而尽，顺势将脑袋抬起来，盯着浅黄色的天花板发呆。正兀自发呆着，精神恍惚小宝被一股淡淡的茉莉花香拉回了图书馆，循着香味，他将脑袋慢慢地低下。四目相对，香味的源头👧也正好奇地盯着他……

**问：**所以，小宝要如何主动又不失礼貌，深入而不猥琐地了解关于她的一切呢？

**答：**通过小程序“获取会员基础信息”能力，在用户授权后，开发者可以获取用户头像图片、昵称、性别、国家、所在省份、所在市区等信息。

本题干的案例和场景仅为课堂效果定制，纯属虚构，请勿作死模仿。对于恶意获取用户信息或者其他不合理使用等情况，支付宝开放平台有权永久收回该小程序的接口权限。

代码千万条，用户第一条。接入不规范，老师两行泪。



## 产品介绍

获取会员基础信息是支付宝会员开放服务之一，在获得用户授权后，允许开发者获取头像、昵称、性别、国家码、所在省份、所在市区等信息。本功能免费，同学们可以放心使用（别问落地价，因为爱情无价）。

当然，小程序还有 **获取会员手机**

号 (<https://opendocs.alipay.com/mini/introduce/getphonenumber>) 的开放能力，因该功能涉及用户的手机号隐私，且仅开放给有一定资质的企业账户，故在本章不作接入介绍。（小宝哭晕在厕所☹️……）

## 用户端示例

用户在登录小程序后，在需要用户授权基础信息的场景（如首次登录，或者授权其他关联账号登录等），系统出现弹窗让用户确认，用户同意授权后，即可通过接口获取用户的基础信息。



**注意：**

通过用户信息授权方式获取用户基础信息是只一种快捷的填写方式。开发过程中，需要对用户拒绝的情况做充分的考虑与应对方案，如引导用户手动填写或上传。

## 缘生：模板消息

因为请对面的女生帮忙一起测试“获取会员基础信息”的缘故，小宝无意间得到了对面女生的昵称和头像。

“原来你叫小美啊，你的支付宝头像是莫奈的《日出》吗？你喜欢画画？”

“对呀，我是设计学院的，平时喜欢画画。看不出你个程序员还知道莫奈啊。”

“我也很喜欢印象派。但我觉得你更适合莫奈的另外一幅画。”

“哦，什么画呀？”

“《睡莲》☺”

**问：**看得出小宝已经词穷了，再聊下去就很尴尬了，看得出小宝也是个害羞人。请问小宝要如何进一步拉近关系，约小美出去自习和玩耍呢 🧑🧑？

**答：**通过小程序“模板消息”能力，开发者可通过消息高效触达用户，通知用户当前行为的结果及状态等；同时可在消息中配置跳转小程序指定页面地址，当用户查看消息时，在消息中点击 **进入小程序查看** 返回小程序，进入开发者配置的小程序指定页面。目前仅支持文本消息。

本题干的案例和场景仅为课堂效果定制，纯属虚构，请勿作死模仿。支付宝开放平台对于模板消息的发送频率和内容均有一定限制。对于恶意发送违规定模板消息的行为，支付宝会有有一定的惩罚策略，详细注意事项可以参考 [模板消息准入条件](#)。代码千万条，用户第一条。接入不规范，老师两行泪。

## 产品介绍

模板消息有两类：**交易类** 和 **表单类**，暂不支持自定义标题等信息。支付类的模板消息需要依赖用户支付产生的交易号 tradeNo，因此本章节不重点阐述，好奇宝宝请前往本章结尾的“拓展阅读”查看。

自己不开店，没有交易号（tradeNo）的小宝别无选择，只能选择表单类模板消息。

**表单类：**当用户在小程序内发生过提交表单行为，开发者可以调用接口发送表单类的模板消息，此时必须要传入 form\_id。开发者获取 formId 或 tradeNo 后，可在 7 天内向用户推送有限条数的模板消息（1 次提交表单可下发 3 条，不限制模板数）；超期后 formId 或 tradeNo 将失效，无法推送消息。

本功能免费，同学们可以放心使用（别问落地价，因为爱情无价）。

## 用户端示例

假设小宝在发送模板消息发送成功后，小美可以在支付宝 APP 首页的“服务提醒”处查看消息，如下图所示。



## 快速接入 DEMO

支付宝开放平台还为开发者提供了模板消息 DEMO，开发者可以参

考 <https://opendocs.alipay.com/mini/quick-example/template-message> 文档进行体验和实现快速接入。

## 拓展阅读

除了用 formId 发送模板消息外，支付宝开放平台还支持用 tradeNo 发送模板消息，那如何获取 tradeNo 呢？

tradeNo：当用户完成支付行为时，可以获取 tradeNo（即支付交易号）用于发送交易类模板消息，如 **小程序支付** 中的 `alipay.trade.create` 或 **资金授权** 以及 **当面付** 中的 `alipay.trade.pay` 接口。消息类型为支付类型的只能通过 tradeNo 发送。

上述几个功能的对接文档在此列出，感兴趣的同学可以课后查看哦。需要注意的是，目前这几大功能仅对企业支付宝账号开放；个人账号暂时无法完成对接。

- 小程序支付：<https://opendocs.alipay.com/mini/introduce/pay>
- 当面付：<https://opendocs.alipay.com/open/194/105072/>
- 资金授权：<https://opendocs.alipay.com/mini/introduce/pre-authorization>

## 缘遂：小程序二维码

“现在到处都是二维码，逼死强迫症哦。忍不住想扫扫看，哈哈。”

“我这边也有个码，你扫了试试？”

“什么二维码啊，这么神秘。扫开了还加载这么久。”

“.....”

“啊，谢谢你。”

夕阳照在小美的脸上，越发得红了。



## 产品介绍

### 概述

使用本能力可生成小程序二维码，商户可将生成后的小程序二维码在线上线下进行贴码推广，更便捷地推广小程序。

### 产品特点

每个小程序都有一个默认的小程序二维码，目标地址是 [小程序首页](#)。

- 创建 20 个以内小程序二维码。在 **开发中心 > 小程序应用 > 我的小程序** 中点击已创建的小程序名称，进入小程序详情页面，左侧目录栏中选择 **码管理 > 小程序码**。
- 创建超过 20 个小程序二维码。通过调用二维码接口 [alipay.open.app.qrcode.create](#) 实现，一个小程序可通过使用该二维码接口获取无限个带参数的二维码。

### 页面地址获取方式

小程序页面地址可通过开发者工具在代码中的 `app.json` 中的 `onShow` 和 `onLaunch` 中获取。

## 使用说明

支付宝扫描二维码将按以下匹配规则控制跳转：

- 页面地址：指定小程序中能访问的路径地址，默认为小程序的首页地址。
- 启动参数：小程序启动时候需要带入的参数，可以为空。启动参数可以通过 `options.query` 获取，格式为 `key1=value1&key2=value2`。

## 示例代码

准备获取启动参数中 `x` 的值。

```
App({
  onLaunch(options) {
    my.alert({content: '启动参数: '+JSON.stringify(options.query),});
    console.log('query', options.query);
    console.log('App Launch', options);
  },
  onShow() {
    console.log('App Show')
  },
  onHide() {
    console.log('App Hide')
  },
  globalData: {
    hasLogin: false
  }
})
```

## 输入参数

页面地址： `page/component/component-pages/view/view`

启动参数： `x=1&y=2`

示例效果：



## 准入条件

小程序开发者均可使用。

## 计费模式

不收费。

## API 列表

| 接口名称  | 描述           |
|---|--------------|
| <a href="#">alipay.open.app.qrcode.create</a> | 小程序生成推广二维码接口 |

## 快速接入 DEMO

支付宝开放平台还为开发者提供了小程序二维码 DEMO，开发者可以参考 [快速示例](#) 文档进行体验和实现快速接入。



# Chapter 4

## 第四章 支付宝小程序快速示例



# 获取手机验证码快速示例

此为简约版获取手机验证码组件，开发者可作为参考，也可直接集成使用。您可以通过 [获取手机验证码快速示例](#) 进行学习操作。

## 扫码体验



## 前提条件

- 已完成 [开发者入驻](#) 与 [小程序创建](#)。
- 已下载并安装 [小程序开发者工具](#)。

## 页面使用

page.xml

```
<mobile-code
  onSendCode="onSendCode"
  onCodeInput="onCodeInput"/>
```

## 组件传参说明

| 字段名              | 简介                                | 类型          | 默认值  |
|------------------|-----------------------------------|-------------|------|
| mobile           | 手机号                               | String      |      |
| verifyCodeLength | 验证码长度                             | Number      | 6    |
| numberCode       | 验证码是否只能输入数字                       | Boolean     | true |
| codeTime         | 验证码倒计时                            | Number      | 60   |
| onSendCode       | 点击发送 重新发送验证码时触发onSendCode(Object) | EventHandle |      |

|             |                            |             |       |
|-------------|----------------------------|-------------|-------|
|             | Object: { mobile }         |             |       |
| disabled    | 手机号是否可以编辑                  | Boolean     | false |
| onCodeInput | 验证码输入时事件<br>onCodeInput(e) | EventHandle |       |
| className   | 自定义组件最外级 class             | String      |       |

# 智能售货柜小程序快速示例

提供智能售货柜小程序模板源码，包含了会员注册\登陆、代扣授权、扫码开门、支付结算、优惠券等功能。您可以通过 [智能售货柜小程序快速示例](#) 进行学习操作。

## 使用说明

- 本示例为纯客户端代码，可直接在模拟器和在真机预览。
- 部分页面暂不支持调试，如遇此类不支持页面，请在 **web IDE > 模拟器 > 页面路径**，切换页面使用。
- 更多使用详情请参见 [代码模板市场](#)。

## 前提条件

- 已完成 [开发者入驻](#) 与 [小程序创建](#)。
- 已下载并安装 [小程序开发者工具](#)（简称 IDE）。

## 使用步骤

1. 打开 IDE 相关的内容目录，关联已有 APPID。
2. 通过 IDE 真机预览。

## 页面内容

本项目包括的页面数量为 11 个，分别是：首页、先享后付首页、优惠券页面、附近设备展示、登录注册页面、个人中心、我的订单、订单详情、订单完结、故障报修、扫码支付等待页面。

### 1. 首页

页面路径：pages/index

包含先享后付和先付后享两个行动点。



## 2. 先享后付首页

页面路径：pages/home

- 优选推荐模块
- 扫一扫功能模块
- 搜索附近设备模块



## 3. 优惠券页面

页面路径：pages/coupon

已使用/未使用两个 tab 优惠券列表。



## 4. 附近设备展示

页面路径：pages/near

- 设备地图定位功能
- 附近设备信息列表



附近无设备

高德地图 © 2019 AutoNavi - GS(2018)1709号

## 5. 登录注册页面

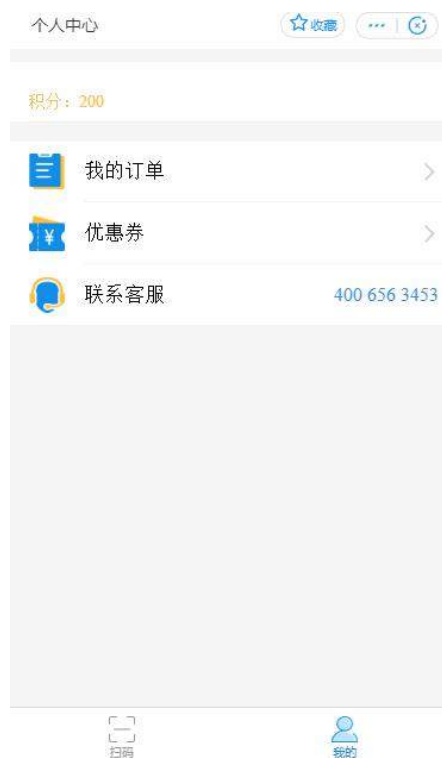
页面路径: pages/registLogin

登录注册功能。

## 6. 个人中心

页面路径: pages/registLogin

- 个人信息展示
- 其它行动点列表



## 7. 我的订单

页面路径: pages/myOrder

我的订单列表。



## 8. 订单详情

页面路径: pages/orderDetails

订单详情信息。



## 9. 订单完结

页面路径：pages/myOrder

订单支付结果页,包含订单信息，优惠券,实付金额，备注等展示功能。



## 10. 故障报修

页面路径：pages/myOrder

- 常用故障列表展示
- 故障信息提交
- 联系客服行动点



## 11. 扫码支付等待页面

页面路径：pages/openDoor

- 等待状态展示
- 支付成功与失败等后续行动点展示

## 自定义 UI 组件

### 1. 优选推荐

页面路径：module/good-list

优选货品推荐展示列表组件。



### 2. 已选商品展示

页面路径：module/good-show

可展示商品详情、金额合计。

### 3. 移动端登录框

页面路径：module/login-form。

可实现手机验证码登录功能。





# Chapter 5

## 第五章 小程序协同工作及版本管理



# 小程序成员管理

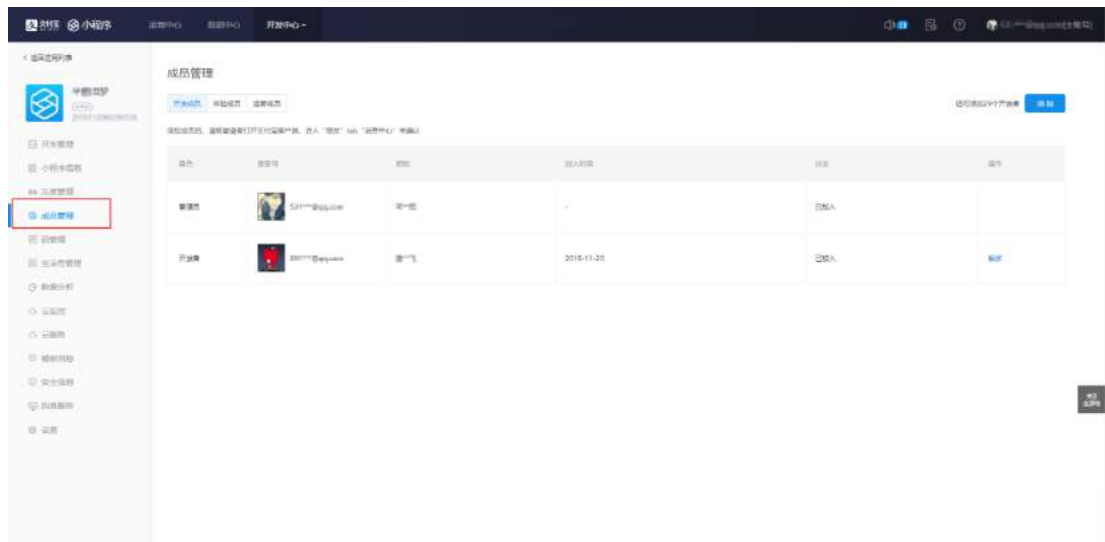
支付宝小程序提供了 **开发成员**、**体验成员**、**运营成员** 三种小程序成员角色，帮助您更好地开发管理支付宝小程序。

## 成员管理权限说明

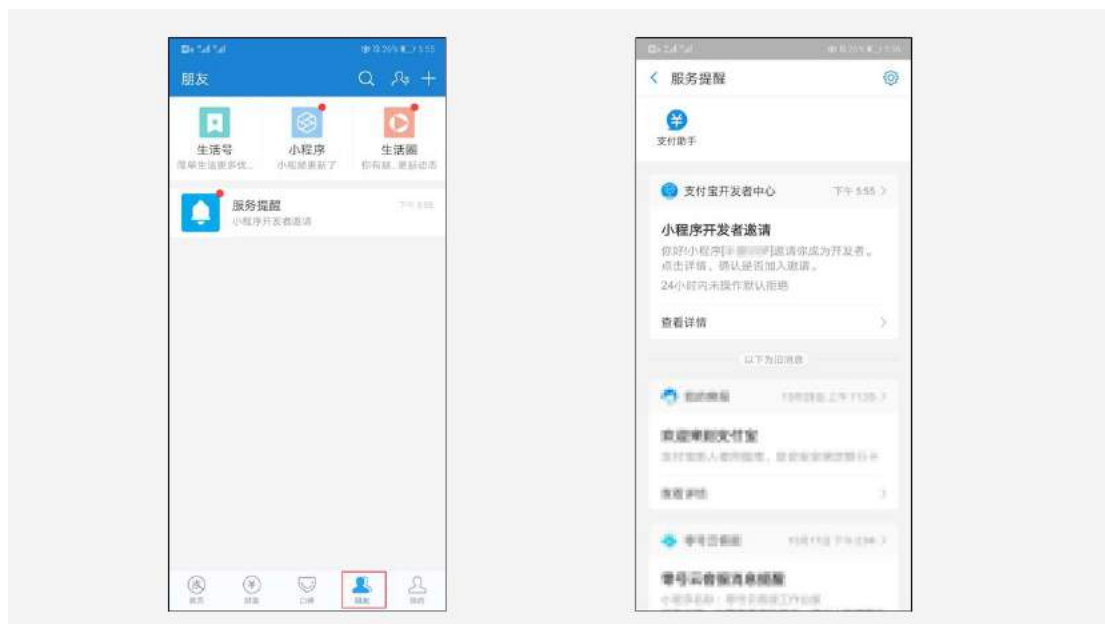
|                   | IDE 内<br>开发调<br>试 | 未上线<br>前预览 | 上传代<br>码 | 提交审<br>核 | 营销运<br>营 | 数据分<br>析 | 删除下<br>架 |
|-------------------|-------------------|------------|----------|----------|----------|----------|----------|
| 开发成<br>员（管<br>理者） | ✓                 | ✓          | ✓        | ✓        | ✓        | ✓        | ✓        |
| 开发成<br>员（开<br>发者） | ✓                 | ✓          | ✓        |          |          |          |          |
| 体验成<br>员          |                   | ✓          |          |          |          |          |          |
| 运营成<br>员          |                   |            |          |          | ✓        | ✓        |          |

## 成员管理入口

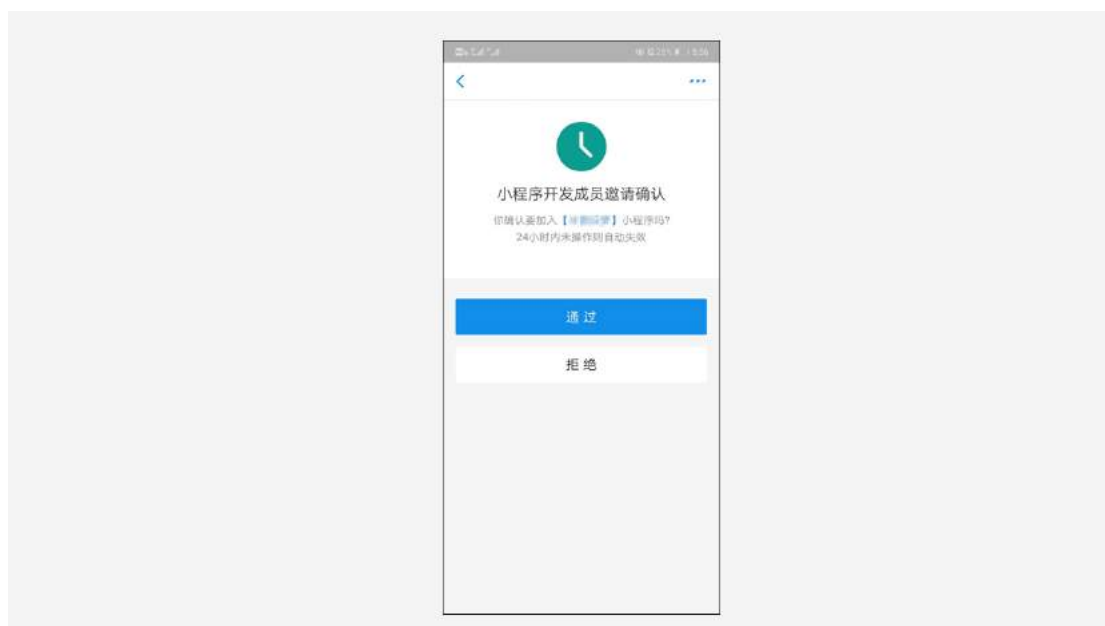
您可以通过 [开发者入驻](#) 并 [创建小程序](#)，在 [我的小程序](#) 中点击 **查看**，选择左树目录上的 **成员管理**，查看到小程序成员管理信息。点击 **添加** 可以搜索添加对应账号为成员。



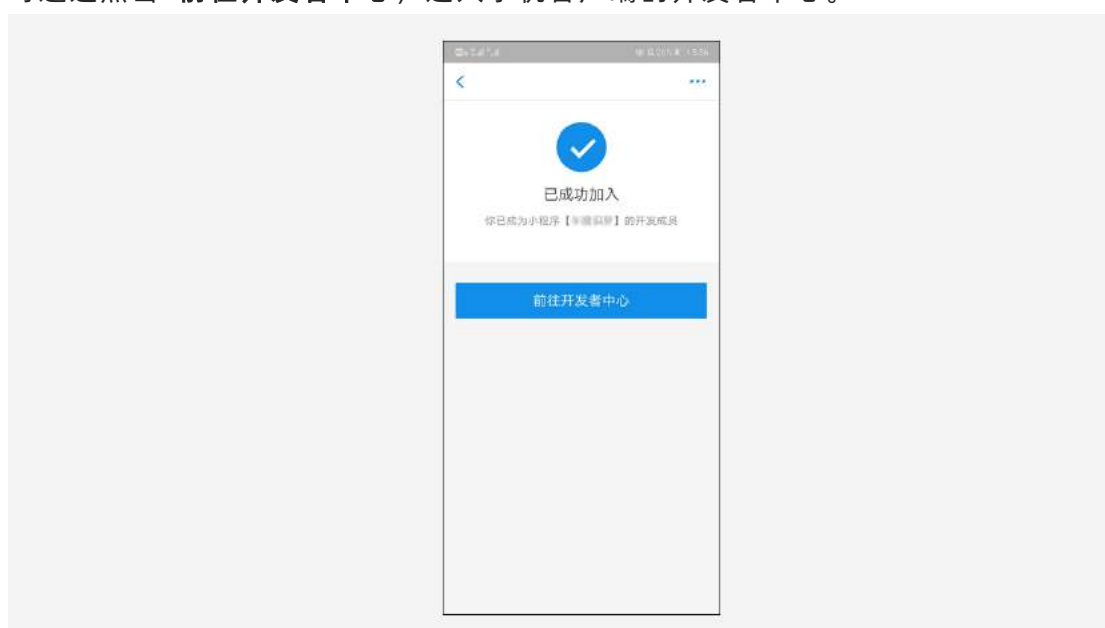
被邀请的成员可以在手机支付宝客户端 **朋友** 中查看请求信息。



点击 **通过** ，接受邀请，成为小程序成员。



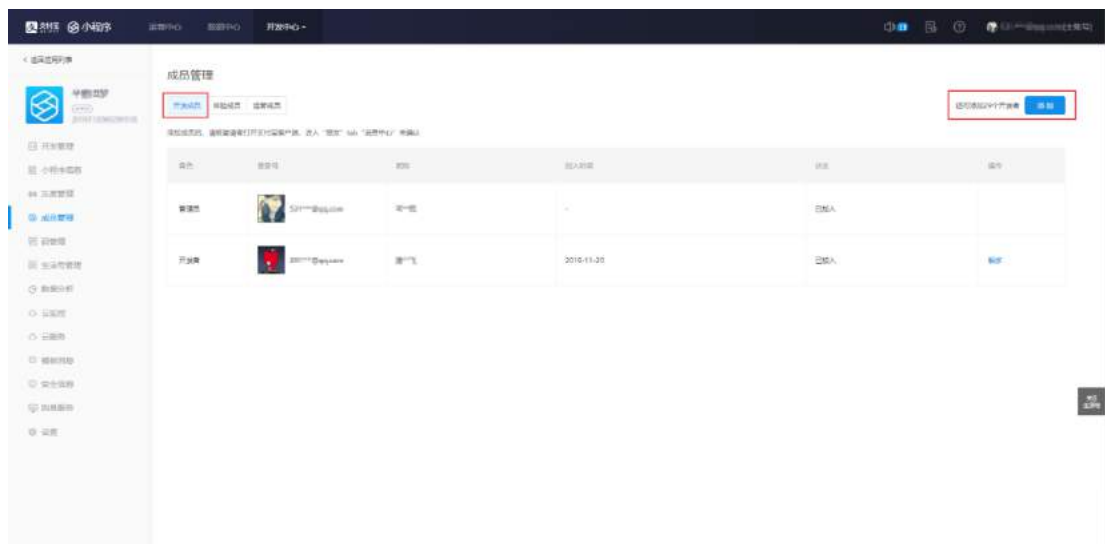
可通过点击 **前往开发者中心**，进入手机客户端的开发者中心。



## 开发成员

支付宝小程序的开发成员包含 **管理员**、**开发者** 两种角色。

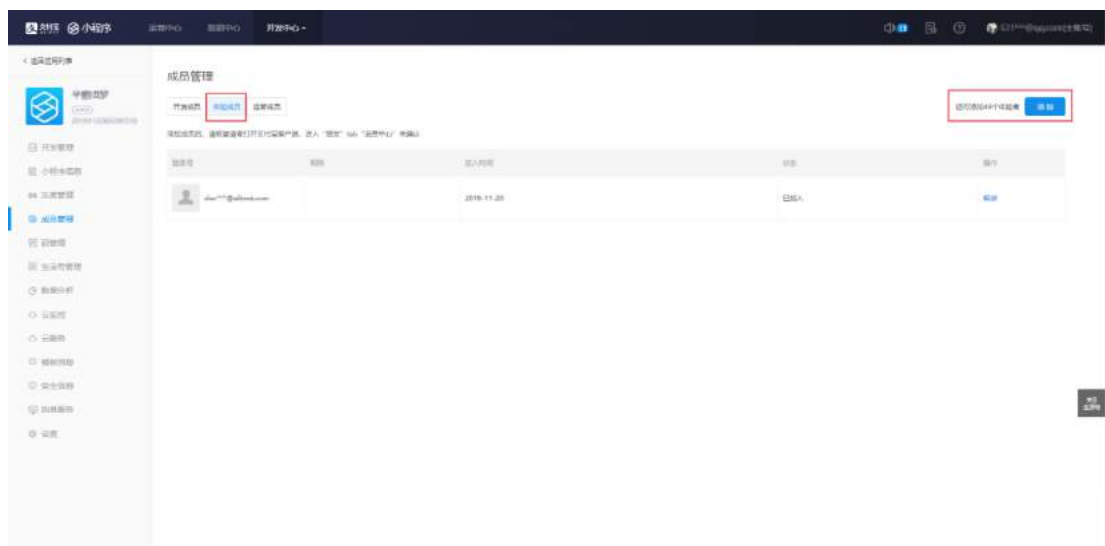
- **管理员** 是创建小程序的账号主体，可以完成 IDE 内开发调试、未上线前预览、上传代码、提交审核、营销运营、数据分析、删除下架 的权限操作。
- **开发者** 是通过接受管理者发出的开发邀请，加入小程序开发的开发人员，可以完成 IDE 内开发调试、未上线前预览、上传代码 的权限操作。
- 一个小程序内最多可以添加 **30** 名开发者。



## 体验成员

支付宝小程序的体验成员可以在小程序还未上线或审核阶段时，率先体验小程序的功能。

- 体验成员可以完成 未上线前预览 的权限操作。
- 一个小程序内最多可以添加 50 名体验成员。

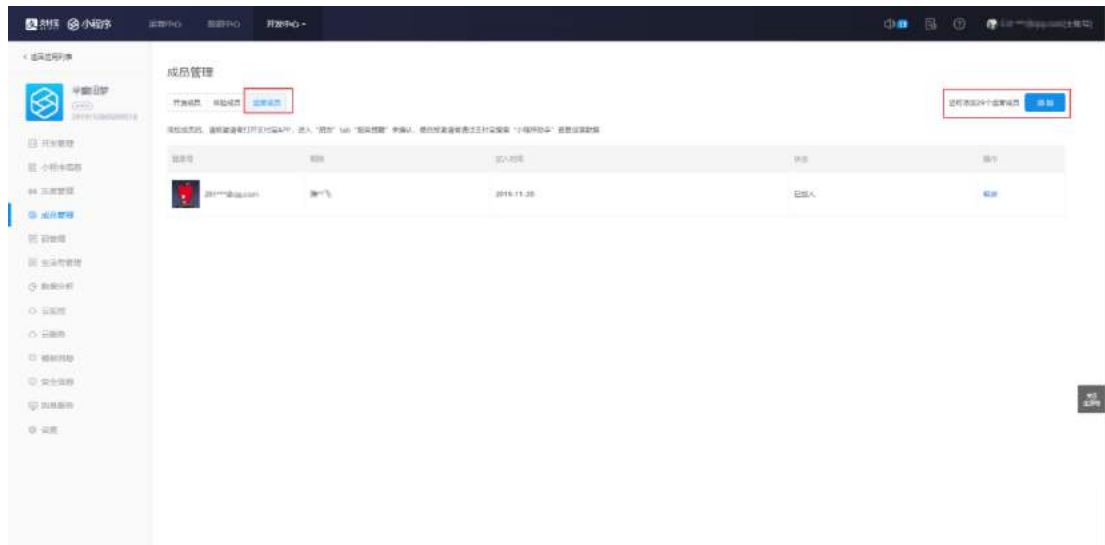


## 运营成员

支付宝小程序的运营成员负责小程序上线后的日常维护、营销运营，并且可以持续为小程序的用户提供服务。

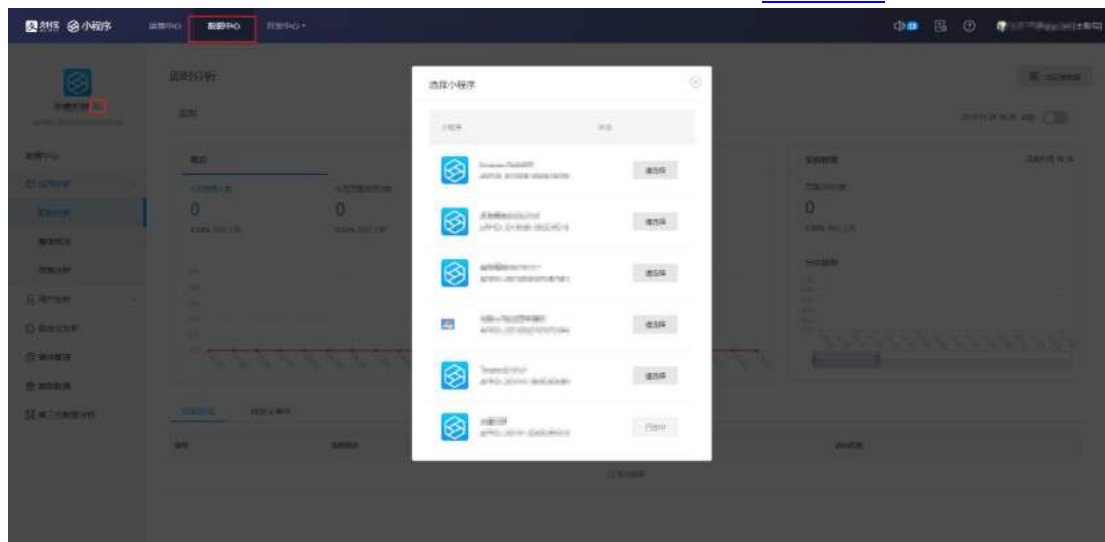
- 运营成员可以完成 营销运营、数据分析 的权限操作。

- 一个小程序内最多可以添加 30 名运营成员。



## 数据分析

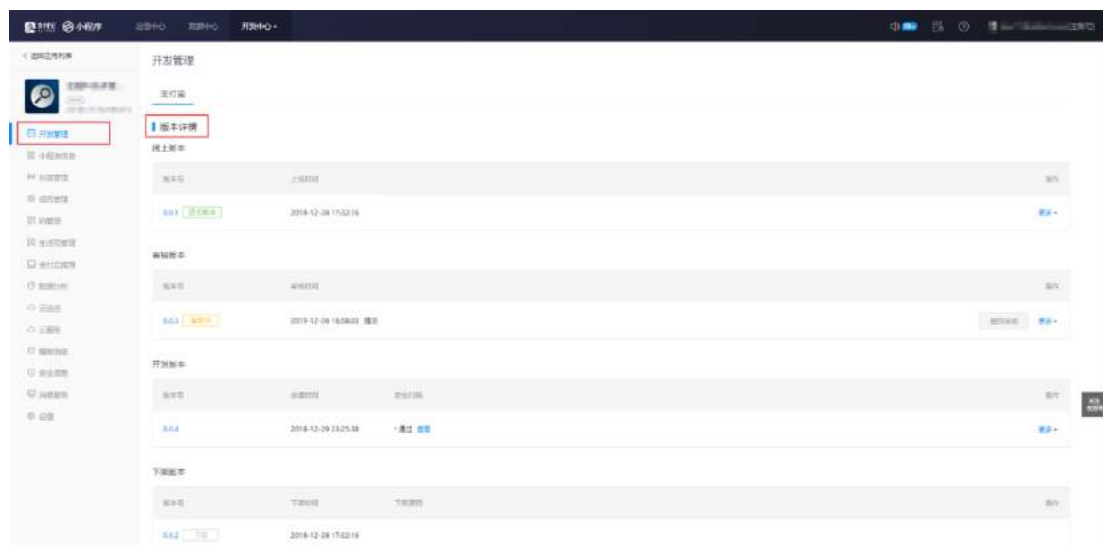
开发成员（管理员）和 运营成员 可以在 [小程序列表页](#)，点击 **数据中心**，选择小程序，对小程序进行数据分析操作。更多信息，请参见 [数据文档](#)。



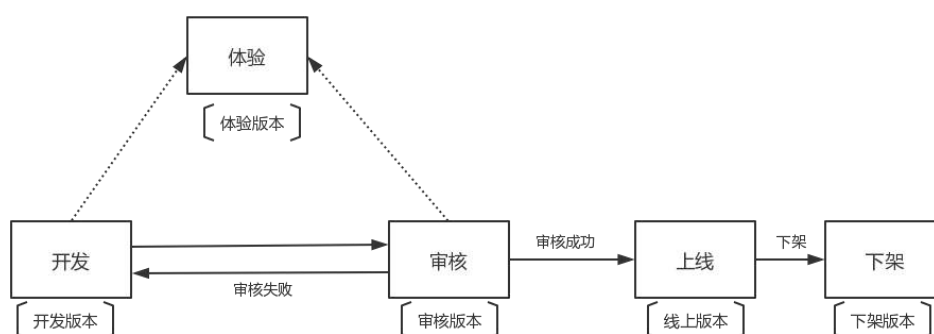
# 小程序版本管理

## 介绍

支付宝小程序的版本主要分为 **开发版本**、**体验版本**、**审核版本**、**线上版本**、**下架版本** 五种版本。您可以在 [我的小程序](#) 中点击 **查看**，在 **开发管理** 里查看小程序的 **版本详情**。



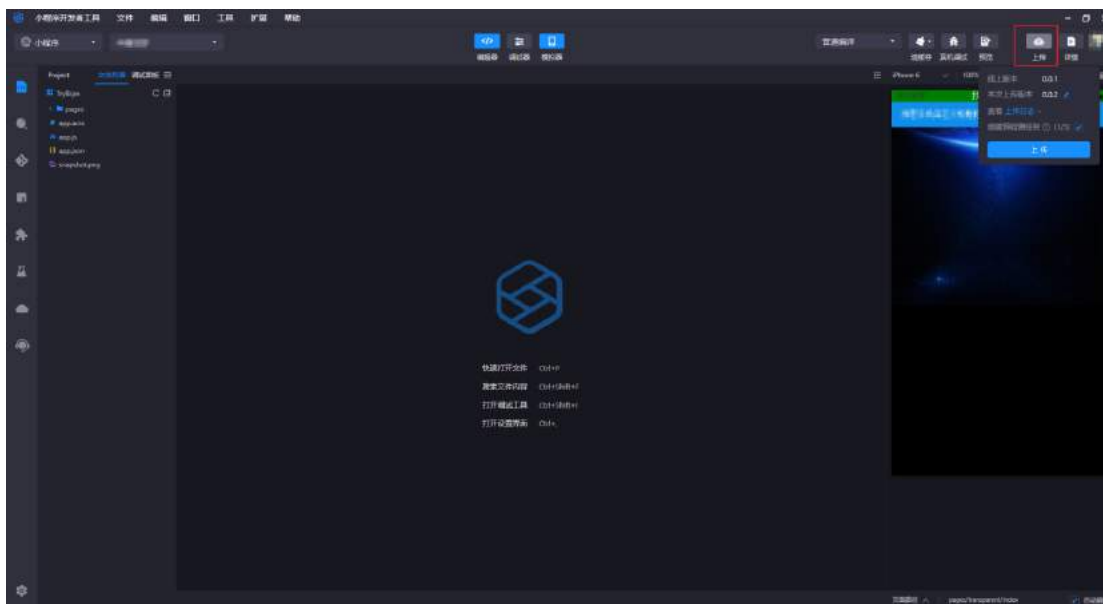
下图展示了小程序整个生命周期中不同阶段对应的版本：



## 开发版本

小程序的 **开发版本** 指的是小程序在 IDE 内开发，上传代码生成以版本号区分的开发版本。

上传代码版本时可编辑本次上传版本号，本次上传版本号必须高于当前开发最新版本；且必须满足 x.y.z 格式，均为数字。



开发版本在上传代码时，会进行安全扫描，上传成功的开发版本可以在 **版本详情** 的 **开发版本** 里看到安全扫描的状态，也可以点击 **查看**，查看到安全扫描内容的详情。

开发版本

| 版本号   | 创建时间                | 审核状态                  | 操作   |
|-------|---------------------|-----------------------|--|
| 0.0.4 | 2018-12-29 23:25:38 | 通过 <a href="#">查看</a> | <a href="#">提交审核</a> <a href="#">更多 &gt;</a> |
| 0.0.3 | 2018-12-29 23:21:49 | 通过 <a href="#">查看</a> | <a href="#">提交审核</a> <a href="#">更多 &gt;</a> |

您可以点击 **提交审核**，将开发版本转成审核中版本。

您也可以点击 **更多 > 设为体验版**，将开发版本设置为体验版。

您还可以直接点击 **更多 > 删除**，删除开发版本，不会影响到 **上架版本** 和 **体验版本** 中的代码。

开发版本

| 版本号   | 创建时间                | 审核状态                  | 操作  |
|-------|---------------------|-----------------------|---|
| 0.0.4 | 2018-12-29 23:25:38 | 通过 <a href="#">查看</a> | <a href="#">提交审核</a> <a href="#">更多 &gt;</a><br><a href="#">设为体验版</a> |
| 0.0.3 | 2018-12-29 23:21:49 | 通过 <a href="#">查看</a> | <a href="#">提交审核</a> <a href="#">更多 &gt;</a><br><a href="#">删除</a>    |

## 体验版本

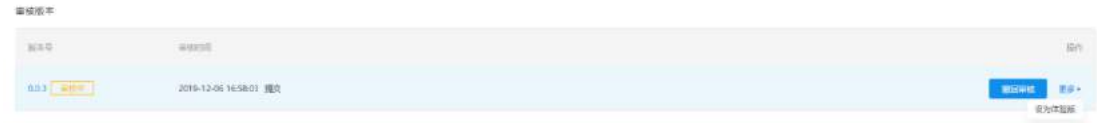
小程序的 **体验版本** 是指将小程序的某个 **开发版本** 或者 **审核中版本** 设置成为体验版，可供小程序的开发成员（管理员）和运营成员进行体验操作。更多信息可参见 [体验版测试](#)。

## 审核版本

小程序的 **审核版本** 是指小程序的 **开发版本** 在点击审核后，处于审核中的版本。只能有一个开发版本处于审核过程中，审核成功后会直接发布成线上版本（若



线上已有版本，将覆盖原线上版本，成为新的线上版本）。您可以 **撤回审核**，也可以点击 **更多 > 设为体验版**，将审核版本设置为体验版。



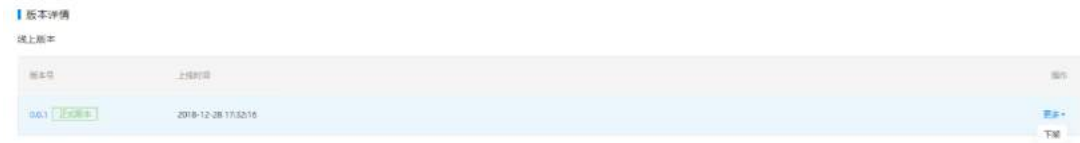
审核失败时，您可以通过点击 **退回开发**，重新提交审核。



## 线上版本

小程序的 **线上版本** 是指通过审核后的代码版本可提供给所有用户使用的小程序版本。

您可以通过点击 **更多 > 下架**，将线上版本下架成下架版本。



## 下架版本

小程序的 **下架版本** 是指线上版本下架之后的版本，您可以点击 **查看全部下架版本** 来查看下架版本信息。



# Chapter 6

## 第六章 小程序基础库更新迭代



# 基础库更新迭代

## 基础库

### 基础库与客户端的关系

基础库是负责小程序框架的加载的容器，提供小程序框架需要的标准组件和标准 API 接口。小程序能力需要支付宝客户端来支撑，每一版基础库新增能力都需要特定版本以上客户端才能运行，高版本基础库的某些新能力无法兼容低版本客户端，关于基础库兼容方法，可以查看兼容章节。可以通过 `my.SDKVersion` 查看当前基础库版本号。

### 基础库更新时机

当基础库准备更新时，会在客户端内进行逐步灰度直到全量发布。当用户客户端更新至最新基础库后，小程序就会运行在最新的基础库上。

随着基础库的不断更新，老版本客户端不支持的能力越来越多，所以基础库支持范围有一个最低客户端版本的要求，即部分老版本客户端以后将无法更新到最新的基础库，会停留在某一历史版本。例如，如果用户的支付宝客户端版本是 **1.0.0** 版本，那么基础库最多更新到 **1.1.0**。

### 基础库版本分布

更新时间：2019 年 12 月 4 日

| 基础库版本    | 用户占比   | 支付宝客户端最低版本 |
|----------|--------|------------|
| >=1.21.4 | 93.64% | 10.1.70    |
| 1.20.5   | 0.79%  | 10.1.68    |
| 1.20.1   | 0.33%  | 10.1.68    |
| 1.19.4   | 0.53%  | 10.1.65    |
| 1.16.4   | 0.35%  | 10.1.58    |
| 1.13.15  | 0.98%  | 10.1.32    |
| 其它       | 3.38%  | -          |

# 设置最低基础库版本

从 [开发中心](#) > 小程序应用 > 我的小程序，进入小程序详情页，点击左侧栏 **设置**，在 **基础设置** 栏，可设置小程序的最低基础库版本。若小程序用户使用的基础库版本低于设置的最低版本要求，则无法正常使用小程序，并将提示用户更新支付宝版本。设置版本号后，小程序需重新发版才会生效。



点击 **最低基础库版本** 对应的 **设置** 按钮，可看到不同的最低基础库版本对应的受影响用户 UV 占比，即近 30 天内访问小程序的用户的基礎庫版本小于所选版本的占比。开发者可据此设置小程序的最低基础库版本。

设置最低基础库版本

若用户使用的基礎庫版本低于设置的最低版本要求，则提示更新支付宝版本。[点此查看基础库更新日志](#)

| 版本                           | 受影响用户UV占比 |
|------------------------------|-----------|
| <input type="radio"/> 1.9.8  | 0.01%     |
| <input type="radio"/> 1.9.7  | 0.0%      |
| <input type="radio"/> 1.9.6  | 0.0%      |
| <input type="radio"/> 1.16.2 | 83.46%    |
| <input type="radio"/> 1.16.1 | 5.41%     |

< 1 2 3 4 5 6 7 8 9 >

5条/页

跳至 1 页

取消

确定

选中需设置为最低基础库版本的版本号，点击 **确定** 按钮，即设置成功。

设置最低基础库版本



若用户使用的基础库版本低于设置的最低版本要求，则提示更新支付宝版本。[点此查看基础库更新日志](#)

| 版本                                      | 受影响用户UV占比 |
|---|-----------|
| <input type="radio"/> 1.14.1            | 0.01%     |
| <input type="radio"/> 1.14.0            | 0.0%      |
| <input type="radio"/> 1.13.9            | 0.02%     |
| <input type="radio"/> 1.13.8            | 0.0%      |
| <input checked="" type="radio"/> 1.13.7 | 0.0%      |

< 1 2 3 4 5 6 7 8 9 >

5 条/页

跳至 1 页

取消

确定

此时，最低基础库版本 设置项对应的 状态 变为 已设置。

设置

| 设置项     | 状态         | 说明   | 操作 |
|---------|------------|--|----|
| 小程序强制跳转 | 已允许所有小程序跳转 | 允许其它小程序跳转到您的小程序  | 修改 |
| 商家可以设置  | 已设置        | 设置是否允许用户通过商家绑定小程序  | 关闭 |
| 最低基础库版本 | 已设置        | 若用户使用的基础库版本低于设置的最低版本要求，将无法正常使用小程序，并提示更新支付宝版本；设置版本越高，需要更新支付宝版本越高。         | 修改 |
| 智能客服    | 未开通        | 开通智能客服后，可在“基础设置”中设置“智能客服”为客服服务方式，在“小程序”-“关于”页面展示客服入口，线上搜索“智能客服”可进入客服工作台。 | 开通 |
| 客服设置    | 未设置        | 设置为用户提供在线客服方式，推荐使用“智能客服”，便于和用户在线实时交流。                                    | 设置 |

# Chapter 7

## 第七章 IoT 小程序开发及刷脸支付实现



# IoT 小程序开发及刷脸支付实现

## 什么是 IoT?

IoT (The Internet of Things) 即物联网，是互联网基础上的延伸和扩展的网络，将各种信息传感设备与互联网结合起来而形成的一个巨大网络，实现在任何时间、任何地点，人、机、物的互联互通。

而 IoT 小程序则是支付宝实现物物相连的一种方式，针对 IoT 的概念，支付宝开发了一系列 IoT 设备，以 IoT 设备为硬件基础，承载 IoT 小程序，实现刷脸购物和支付等功能。

## IoT 小程序简介

从设备角度上说，IoT 小程序也是一种实现 IoT 设备二次开发的方法。类似支付宝小程序，IoT 小程序开放了一系列的 API 和组件。开发者可以快速开发一个 IoT 小程序，定制 IoT 设备功能，满足各行业个性化的需求。

## IoT 设备

目前支付宝已经开发了一系列 IoT 的产品，详细产品介绍请至支付宝 IoT 官方网站 (<https://iot.open.alipay.com>) 查看。目前已经支持开发 IoT 小程序的主要是蜻蜓系列设备，目前包括：支付宝盒-F1、支付宝盒-F4、蚂里奥刷脸设备 T1A、商米刷脸设备 T3B00、支付宝盒 F4H、支付宝盒 F4 plus。如下图所示：



蜻蜓设备专为 IoT 设计，配备前置摄像头，提供刷脸功能，可以通过接入 IoT 小程序实现更多便捷的功能。

## IoT 小程序与多端发布

### 什么是多端?

所谓多端，其实就是指不同的应用终端，支付宝小程序是一种应用终端，IoT 小程序也是一种应用终端，另外还有淘宝、钉钉等开发了小程序就又是另外的应用终端。

那么多端发布则是指使用支付宝小程序提供的各种接口，可以开发其他端的小程序并发布到各个端使用。也就是说开发者开发一次支付宝小程序，同一套代码不做任何改动就可以通过接口发布到各个端，同时触达更多的用户群体，节约了开发成本。

支付宝小程序目前支持的终端有：



所以 IoT 小程序的开发也可以使用支付宝小程序的接口进行开发。



# Chapter 8

## 第八章 小程序模板及插件开发应用



# 小程序模板介绍

## 角色介绍

- 系统服务商（ISV）：负责开发、维护小程序模板，基于模板实例化出商户小程序。是被委托方。
- 商户：委托系统服务商（ISV）基于系统服务商（ISV）提供的小程序模板实例化出商户自己的小程序。是委托方。

## 模板开发模式的核心

三方小程序模板，是支付宝为系统服务商（ISV）提供批量生成小程序的能力，系统服务商（ISV）通过模板开发，得到商户授权后，即可快速根据模板代码给商户快速实例化小程序。

模板开发模式下系统服务商（ISV）可以通过 API 接口批量处理商家小程序，比如系统服务商（ISV）通过模板开发模式服务了 1 万个不同的餐饮小程序，如果需要对这 1 万个不同的小程序做版本升级时，在模板开发模式下，可以实现用 API 接口构建、提审、上架这 1 万个不同的小程序。

在模板开发模式下商户把自己的小程序授权给系统服务商（ISV）的三方应用，让系统服务商（ISV）去帮商户实现商户小程序的代码构建、提审以及商户小程序上架。在商户把商户小程序授权给系统服务商（ISV）的三方应用后，商户小程序的代码只能通过服务商三方应用下的小程序模板代码去构建。

### 业务特点：

- 开发流程：系统服务商（ISV）先开发小程序模板的代码，然后通过小程序模板的代码去构建商户小程序的代码。
- 快速开通：通过服务市场，商户仅需完成订购，即完成商户小程序授权，通过模板快速生成自有商户小程序。
- 批量维护：系统服务商（ISV）可代商户实现小程序的快速批量更新。

## 模式简介

系统服务商（ISV）可以通过研发小程序模板的方式，为商户提供标准化的小程序代开发服务，快速规模化的拓展商户。

模板开发主要有四步：

- 1、系统服务商（ISV） 需创建和开发第三方应用。如已经拥有，则可以沿用；
- 2、在第三方应用下创建小程序模板，并完成开发；
- 3、引导商户完成小程序授权：
  - 系统服务商（ISV）通过第三方应用，引导商户将小程序应用授权给自己，以便进行后面模板小程序实例化的实施操作；
  - 系统服务商（ISV）发布模板为服务，并引导商户在服务市场订购对应小程序模板，或完成小程序应用授权。操作如下：
    - 系统服务商（ISV） 引导商户在服务市场订购小程序模板。如商户没有小程序，在订购过程中平台会让商户完成小程序的创建并授权；
    - 系统服务商（ISV）通过线下推广的方式，引导商户完成模板服务的授权。如商户没有小程序，在授权过程中平台会让商户完成小程序的创建；
- 4、系统服务商（ISV）通过平台接口完成模板小程序的构建、提审、测试、上架等操作。

# 小程序插件介绍

插件是小程序的一项能力，插件 2.0 体系使插件不仅像组件一样可以嵌入到主体小程序页面内部，而且还可以像页面一样提供独立服务，可以从主体小程序进行跳转，同时我们对主体小程序和插件进行权限隔离，在保证插件数据安全的前提下提供更多的灵活性。目前只有 **支付宝官方** 才能开发，但所有的开发者可以通过支付宝服务市场来获取和使用这些插件。

每个插件有插件 ID（和小程序 AppID 同一命名空间），形态在功能和结构上和小程序类似，不同的是插件运行在主体小程序的上下文之中，对主体小程序的能力进行扩展。

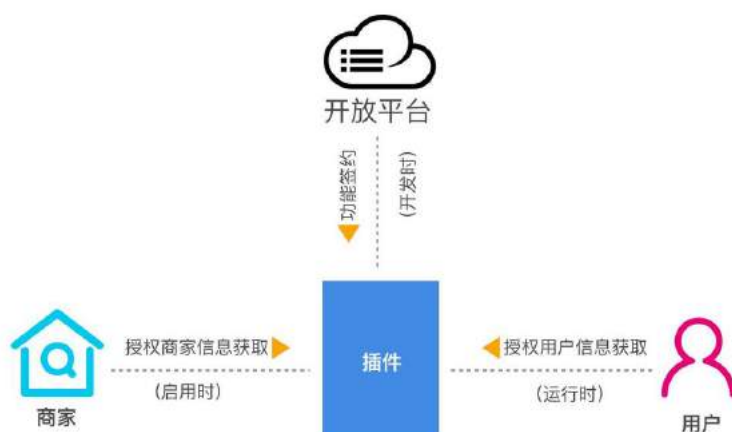
目前插件主要聚焦于门店小程序的场景，用于三方开发者对各门店进行能力上的扩展，如电子发票、配送和预订等。

## 主体小程序和插件的区别

|       | ID                         | 运行时 AppID       | 开发者         | 消费者         |
|-------|----------------------------|-----------------|-------------|-------------|
| 主体小程序 | 小程序 APPID                  | 小程序 AppID       | 主题小程序开发者    | 主题小程序用户     |
| 小程序插件 | 插件 ID（和小程序 AppID 共用同一命名空间） | 和主体小程序 AppID 绑定 | 独立于主体小程序开发者 | 实体小程序开发者和用户 |

## 插件与开放平台、商家及用户之间的关系

- 在开发时，插件开发商需要和开放平台签约获得特色能力（如支付）。
- 商家购买插件后，在门店启用时，根据插件所需要获得的信息（如门店名称）对插件进行授权，这样插件在运行时可获取被授权的信息。
- 在用户使用过程中，唤起插件时，如果插件需要获取用户信息（如用户名称、最近消费金额和时间），需要用户授权。



## 插件开发和使用涉及到的场景

### 场景 1：小程序及小程序模板开发

- 小程序和小程序模板（特别是后者）在开发时，需要明确是否支持插件热插拔，如果需要支持，需预留可插拔的位置，需要定义相关的参数传递处理逻辑，例如对于插件所需要的不同参数的获得与传递。
- 在发布时，需要明确自己将对插件暴露什么样的参数 id、类型和描述，这样插件开发者可以通过这些约定来获得这些参数。

### 场景 2：插件开发

- 插件的业务目标决定了它被唤起时需要获得主体程序什么样的信息，这个时候需要定义插件期待的信息参数 id：例如对于电子发票小服务，它希望在唤起的时候能够获得主体小程序当前门店的名称。
- 另外对于插件所需要的功能包，需要和开放平台进行签约。

### 场景 3：插件的购买和启用

- 商家在选择购买和启用插件的时候，如果插件需要获得商家小程序的信息，需显示出来并让商家进行明确的授权：例如对于电子发票插件，订购和启用时，会提示该插件需要在运行时获得当前门店的名称，这需要商家明确同意，否则不能购买和启用。
- 如果插件需要的信息在商家小程序中并不存在，则表明插件不适用于该商家，不能进行购买和启用。

### 场景 4：插件的使用

- 用户可以在主体小程序唤起插件，例如在门店点击电子发票进行开票。插件如果在这个时候需要获得用户相关的信息如用户名称、当前门店消费金额和时间，需要用户明确授权。
- 对于商家已在购买和启用时所授权的信息如门店名称，主体小程序在唤起插件时会根据插件所需要的参数 id 传递过来相关的信息。
- 插件利用获得的用户信息和商家信息，运行相关业务逻辑对用户提供服务。



阿里云开发者“藏经阁”  
海量免费电子书下载

支付宝小程序文档团队

