# 微信小程序

官方文档：<https://mp.weixin.qq.com/debug/wxadoc/dev/?t=1477656487237>

--简单说，微信“小程序”可以为开发者提供基于微信的表单、导航、地图、媒体和位置等开发组件，让他们在微信的网页里构建一个HTML 5应用。

同时微信还开放了登录和微信支付等接口，让这个“小程序”可以和用户的微信账号打通。

虽然微信小程序本质上来说就是一个HTML 5（移动网页） 应用，但与那些经常在朋友圈刷屏的 H5 小游戏或者应用不同的是，微信小程序获得更多的系统权限。

首先是数据缓存能力，这可以让用户在打开一个小程序的时候将程序的主要框架缓存到微信上，下一次就可以快速打开了。

微信创始人张小龙曾说过，微信应用号希望实现的目标是“用完即走，无需安装和卸载”，也就是说以后当你要使用一个应用时，只需要在微信里搜索就可以直接使用了。

-------------------Google一直希望将Chrome浏览器变成一个超级网页App平台，Google在最近提出了一个名为Progressive Web App的概念，指的是运行在浏览器里的App。

这类App跟微信小程序的原理很类似，也就是将应用需要的框架和进程缓存到本地的浏览器里，让H5程序在离线的状态下仍然能够使用。

## 获取微信小程序的 AppID 开发工具

### 安装开发环境

#### 下载地址

- [Windows 64位]

(https://servicewechat.com/wxa-dev-logic/download\_redirect?type=x64&from=mpwiki&t=1475052055457)

- [Windows 32位]

(https://servicewechat.com/wxa-dev-logic/download\_redirect?type=ia32&from=mpwiki&t=1475052055457)

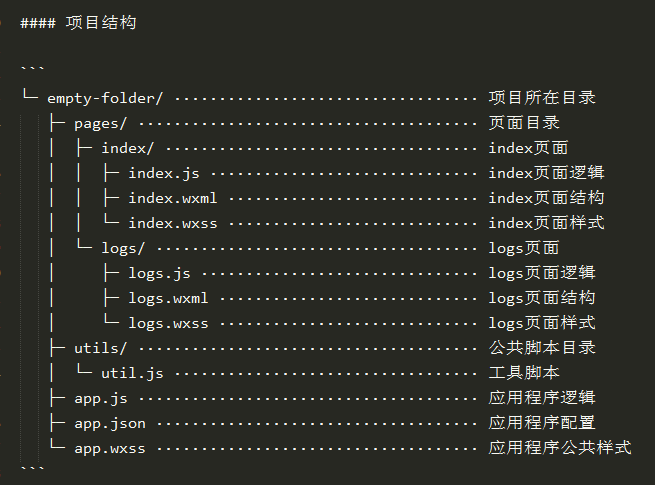
- [macOS]

(https://servicewechat.com/wxa-dev-logic/download\_redirect?type=darwin&from=mpwiki&t=1475052055457)

## 创建项目

借助官方的`QuickStart项目`完成最基本的`Hello world`，

## 项目结构



最关键也是必不可少的，是 app.js、app.json、app.wxss 这三个。

其中，.js后缀的是脚本文件，.json后缀的文件是配置文件，.wxss后缀的是样式表文件。微信小程序会读取这些文件，并生成[小程序实例](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/app-service/app.html?t=1477656485442)。

[page-name].js

页面逻辑文件，用于创建页面对象，以及处理页面生命周期控制和数据处理

[page-name].json

设置当前页面工作时的window的配置，此处会覆盖app.json中的window设置，也就是说只可以设置window中设置的属性

[page-name].wxml

wxml指的是Wei Xin Markup Language

用于定义页面中元素结构的，语法遵循XML语法，注意是XML语法，不是HTML语法，不是HTML语法，不是HTML语法

[page-name].wxss

wxml指的是Wei Xin Style Sheet

用于定义页面样式的，语法遵循CSS语法，扩展了CSS基本用法和长度单位（主要就是rpx响应式像素）

### 项目配置

小程序中的配置文件分为两种：

* 全局配置文件，根目录下的app.json
* 页面配置文件，每个页面目录下的[page-name].json

#### app.json

项目配置声明文件（指定项目的一些信息，比如导航栏样式颜色等等）

{

// 当前程序是由哪些页面组成的（第一项默认为初始页面）

// 所有使用到的组件或页面都必须在此体现

// https://mp.weixin.qq.com/debug/wxadoc/dev/framework/config.html#pages

"pages": [ ... ],

// 应用程序窗口设置

// https://mp.weixin.qq.com/debug/wxadoc/dev/framework/config.html#window

"window": { ... },

// 应用导航栏设置

// https://mp.weixin.qq.com/debug/wxadoc/dev/framework/config.html#tabBar

"tabBar": { ... },

// 网络超时设置

// https://mp.weixin.qq.com/debug/wxadoc/dev/framework/config.html#networkTimeout

"networkTimeout": {},

// 是否在控制台输出调试信息

// https://mp.weixin.qq.com/debug/wxadoc/dev/framework/config.html#debug

"debug": true

}

#### [page-name].json

用于指定特定页面工作时，window的设置：

{

// 导航条背景色

"navigationBarBackgroundColor": "#35495e",

// 导航条前景色（只能是white/black）

"navigationBarTextStyle": "white",

// 导航条文本

"navigationBarTitleText": "电影 « 豆瓣",

// 窗口背景颜色

"backgroundColor": "#fff",

// 窗口前景色

"backgroundTextStyle": "dark",

// 是否开启下拉刷新

"enablePullDownRefresh": true

}

### 逻辑层分析

#### 应用程序逻辑app.js

app.js作为项目主入口文件，用于创建应用程序对象

// App函数是一个全局函数，用于创建应用程序对象App({

// ========== 全局数据对象（可以整个应用程序共享） ==========

globalData: { ... },

// ========== 应用程序全局方法 ==========

method1 (p1, p2) { ... },

method2 (p1, p2) { ... },

// ========== 生命周期方法 ==========

// 应用程序启动时触发一次

onLaunch () { ... },

// 当应用程序进入前台显示状态时触发

onShow () { ... },

// 当应用程序进入后台状态时触发

onHide () { ... }

})

也就是说，当应用程序启动时会自动执行项目目录下的app.js文件。

| 属性 | 类型 | 描述 | 触发时机 |
| --- | --- | --- | --- |
| onLaunch | Function | 生命周期函数--监听小程序初始化 | 当小程序初始化完成时，会触发 onLaunch（全局只触发一次） |
| onShow | Function | 生命周期函数--监听小程序显示 | 当小程序启动，或从后台进入前台显示，会触发 onShow |
| onHide | Function | 生命周期函数--监听小程序隐藏 | 当小程序从前台进入后台，会触发 onHide |

在app.js中通过调用全局App([option])方法创建一个应用程序实例。

其中通过参数指定的一些特定的方法，会在特定的执行时机去执行，也就是说通常所说的生命周期事件方法。

也可以定义任意其他的对象成员（例如：方法和属性），这些成员可以在内部直接使用，或者外部通过获取app对象调用：

| 属性 | 类型 | 描述 |
| --- | --- | --- |
| 其他 | Any | 开发者可以添加任意的函数或数据到 Object 参数中，用 this 可以访问，一般用于存放业务逻辑配置，比如：API地址 |

#### app.js

App({

data1: '123',

data2: { message: 'hello world' },

api: {

list: 'https://github.com/zce/',

detail: 'https://github.com/zce/',

},

foo () {

return 'bar'

}

})

#### other.js

// getApp 也是全局函数，可以在任意地方调用，用于获取全局应用程序实例对象

var app = getApp()

console.log(app.data1)

console.log(app.data2)

console.log(app.foo())

#### 页面逻辑[page-name].js

[page-name].js是一个页面的重要组成部分，用于创建页面对象

1. 增加 [App](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/app-service/app.html?t=20161102) 和 [Page](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/app-service/page.html?t=20161102) 方法，进行程序和页面的注册。
2. 增加 getApp 和 getCurrentPages 方法，分别用来获取 App 实例和当前页面栈。
3. 提供丰富的 [API](https://mp.weixin.qq.com/debug/wxadoc/dev/api/?t=20161102)，如微信用户数据，扫一扫，支付等微信特有能力。
4. 每个页面有独立的[作用域](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/app-service/module.html?t=20161102" \l "文件作用域)，并提供[模块化](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/app-service/module.html?t=20161102" \l "模块化)能力。
5. 由于框架并非运行在浏览器中，所以 JavaScript 在 web 中一些能力都无法使用，如 document，window 等。
6. 开发者写的所有代码最终将会打包成一份 JavaScript，并在小程序启动的时候运行，直到小程序销毁。类似 ServiceWorker，所以逻辑层也称之为 App Service。

// 获取全局应用程序对象

const app = getApp()

// Page也是一个全局函数，用来创建页面对象

Page({

// ========== 页面数据对象（可以暴露到视图中，完成数据绑定） ==========

data: { ... },

// ========== 页面方法（可以用于抽象一些公共的行为，例如加载数据，也可以用于定义事件处理函数） ==========

method1 (p1, p2) { ... },

method2 (p1, p2) { ... },

// ========== 生命周期方法 ==========

// 页面加载触发

onLoad () { ... }

...

})

在应用程序执行到当前页面时，会执行当前页面下对应的[page-name].js文件。

在[page-name].js中通过调用全局Page([option])方法创建一个页面实例。

##### **Page([option])方法[option]参数说明**

| 属性 | 类型 | 描述 |
| --- | --- | --- |
| data | Object | 页面的初始数据 |
| onLoad | Function | 生命周期函数--监听页面加载 |
| onReady | Function | 生命周期函数--监听页面初次渲染完成 |
| onShow | Function | 生命周期函数--监听页面显示 |
| onHide | Function | 生命周期函数--监听页面隐藏 |
| onUnload | Function | 生命周期函数--监听页面卸载 |
| onPullDownRefreash | Function | 页面相关事件处理函数--监听用户下拉动作 |
| 其他 | Any | 开发者可以添加任意的函数或数据到 object 参数中，用 this 可以访问 |

### 视图层分析

#### [page-name].wxml页面结构

WXML（WeiXin Markup Language）是MINA框架设计的一套标签语言，基于XML。

结合一些基础组件、事件系统、模板数据绑定，可以构建出页面的结构。

简单来说：wxml ≈ xml + 事件系统 + 模板引擎

例如：

// js

Page({

data: {

todos: [

{ text: 'JavaScript', completed: false },

{ text: 'JavaScript+', completed: false },

{ text: 'JavaScript++', completed: false }

]

},

completed (e) { ... }

})

<!-- wxml -->

<view>

<view wx:for="{{ todos }}">

<block wx:if="{{ !item.completed }}">

<text>{{ item.text }}</text>

<button bindtap="completed" data-item-index="{{ index }}"> √ </button>

</block>

</view>

</view>

#### [page-name].wxss页面样式

WXSS具有CSS大部分特性。

与CSS相比扩展的特性有：

* 尺寸单位 rpx
* 样式导入（CSS也有）@import

##### **尺寸单位**

rpx（responsive pixel）: 可以根据屏幕宽度进行自适应。

规定屏幕宽为750rpx。如在 iPhone6 上，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。

rem（root em）: 规定屏幕宽度为20rem；1rem = (750/20)rpx 。

| 设备 | rpx换算px (屏幕宽度/750) | px换算rpx (750/屏幕宽度) | rem换算rpx (750/20) |
| --- | --- | --- | --- |
| iPhone5 | 1rpx = 0.42px | 1px = 2.34rpx | 1rem = 37.5rpx |
| iPhone6 | 1rpx = 0.5px | 1px = 2rpx | 1rem = 37.5rpx |
| iPhone6s | 1rpx = 0.552px | 1px = 1.81rpx | 1rem = 37.5rpx |

### 样式导入

使用@import语句可以导入外联样式表，@import后跟需要导入的外联样式表的相对路径，用;表示语句结束。

@import "common.wxss";

text {

background-color: #ff0;

}

### 选择器

目前只支持如下选择器

| 选择器 | 样例 | 样例描述 |
| --- | --- | --- |
| .class | .intro | 选择所有拥有 class="intro" 的组件 |
| #id | #firstname | 选择拥有 id="firstname" 的组件 |
| element | view | 选择所有 view 组件 |
| element, element | view | checkbox 选择所有文档的 view 组件和所有的 checkbox 组件 |
| ::after | view::after | 在 view 组件后边插入内容 |
| ::before | view::before | 在 view 组件前边插入内容 |

### 全局样式与局部样式

定义在app.wxss中的样式为全局样式，作用于每一个页面。

在[page-name].wxss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.wxss中相同的选择器。

## 手机预览

开发者工具左侧菜单栏选择"项目"，点击"预览"，扫码后即可在微信客户端中体验。 在 JavaScript 的基础上，我们做了一些修改

调试工具分为 6 大功能模块：Wxml、Console、Sources、Network、Appdata、Storage