“微信小程序-API”参考文档

（从官网拷贝（2017-04-20），用于没有网络的地方开发时作参考用。）

（非完整版，后面画图、客服等不常用的内容没有拷贝。）

（更多应用请参见：<http://www.cftea.com/weixin/>）

wx.request(OBJECT)

wx.request发起的是 HTTPS 请求。

**OBJECT参数说明：**

| 参数名 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| url | String | 是 | 开发者服务器接口地址 |
| data | Object、String | 否 | 请求的参数 |
| header | Object | 否 | 设置请求的 header , header 中不能设置 Referer |
| method | String | 否 | 默认为 GET，有效值：OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT |
| dataType | String | 否 | 默认为 json。如果设置了 dataType 为 json，则会尝试对响应的数据做一次 JSON.parse |
| success | Function | 否 | 收到开发者服务成功返回的回调函数，res = {data: '开发者服务器返回的内容'} |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数说明：**

| 参数 | 说明 |
| --- | --- |
| data | 开发者服务器返回的数据 |
| statusCode | 开发者服务器返回的状态码 |

**data 数据说明** 最终发送给服务器的数据是 String 类型，如果传入的 data 不是 String 类型，会被转换成 String 。转换规则如下：

* 对于 header['content-type'] 为 'application/json' 的数据，会对数据进行 JSON 序列化
* 对于 header['content-type'] 为 'application/x-www-form-urlencoded' 的数据，会将数据转换成 query string （encodeURIComponent(k)=encodeURIComponent(v)&encodeURIComponent(k)=encodeURIComponent(v)...）

**示例代码：**

wx.request({

 url: 'test.php', //仅为示例，并非真实的接口地址

 data: {

 x: '' ,

 y: ''

 },

 header: {

 'content-type': 'application/json'

 },

 success: function(res) {

 console.log(res.data)

 }

})

Bug & Tip

1. tip: content-type 默认为 'application/json'
2. bug: 开发者工具 0.10.102800 版本，header 的 content-type 设置异常；
3. tip: 客户端的 HTTPS TLS 版本为1.2，但 Android 的部分机型还未支持 TLS 1.2，所以请确保 HTTPS 服务器的 TLS 版本支持1.2及以下版本；
4. tip: 要注意 method 的 value 必须为大写（例如：GET）；
5. tip: url 中不能有端口；
6. tip: request 的默认超时时间和最大超时时间都是 60s
7. tip: request 的最大并发数是 5
8. tip: 网络请求的 referer 是不可以设置的，格式固定为 https://servicewechat.com/{appid}/{version}/page-frame.html，其中 {appid} 为小程序的 appid，{version} 为小程序的版本号，版本号为 0 表示为开发版。

wx.uploadFile(OBJECT)

将本地资源上传到开发者服务器。如页面通过 [wx.chooseImage](https://mp.weixin.qq.com/debug/wxadoc/dev/api/media-picture.html#wxchooseimageobject) 等接口获取到一个本地资源的临时文件路径后，可通过此接口将本地资源上传到指定服务器。客户端发起一个 HTTPS POST 请求，其中 content-type 为 multipart/form-data 。

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| url | String | 是 | 开发者服务器 url |
| filePath | String | 是 | 要上传文件资源的路径 |
| name | String | 是 | 文件对应的 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 |
| header | Object | 否 | HTTP 请求 Header , header 中不能设置 Referer |
| formData | Object | 否 | HTTP 请求中其他额外的 form data |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数说明：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| data | String | 开发者服务器返回的数据 |
| statusCode | Number | HTTP状态码 |

**示例代码：**

wx.chooseImage({

 success: function(res) {

 var tempFilePaths = res.tempFilePaths

 wx.uploadFile({

 url: 'http://example.weixin.qq.com/upload', //仅为示例，非真实的接口地址

 filePath: tempFilePaths[0],

 name: 'file',

 formData:{

 'user': 'test'

 },

 success: function(res){

 var data = res.data

 //do something

 }

 })

 }

})

Bug & Tip

1. tip: 最大并发限制是 10 个
2. tip: 默认超时时间和最大超时时间都是 60s

wx.downloadFile(OBJECT)

下载文件资源到本地。客户端直接发起一个 HTTP GET 请求，返回文件的本地临时路径。

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 必填 |
| --- | --- | --- | --- |
| url | String | 是 | 下载资源的 url |
| header | Object | 否 | HTTP 请求 Header |
| success | Function | 否 | 下载成功后以 tempFilePath 的形式传给页面，res = {tempFilePath: '文件的临时路径'} |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**注：文件的临时路径，在小程序本次启动期间可以正常使用，如需持久保存，需在主动调用**[**wx.saveFile**](https://mp.weixin.qq.com/debug/wxadoc/dev/api/file.html)**，在小程序下次启动时才能访问得到。**

**示例代码:**

wx.downloadFile({

 url: 'http://example.com/audio/123', //仅为示例，并非真实的资源

 success: function(res) {

 wx.playVoice({

 filePath: res.tempFilePath

 })

 }

})

Bug & Tip

1. tip: 最大并发限制是 10 个
2. tip: 默认超时时间和最大超时时间都是 60s
3. tip: 网络请求的 referer 是不可以设置的，格式固定为 https://servicewechat.com/{appid}/{version}/page-frame.html，其中 {appid} 为小程序的 appid，{version} 为小程序的版本号，版本号为 0 表示为开发版。
4. tip: 6.5.3 以及之前版本的 iOS 微信客户端 header 设置无效

wx.connectSocket(OBJECT)

创建一个 [WebSocket](https://developer.mozilla.org/zh-CN/docs/Web/API/WebSocket) 连接；**一个微信小程序同时只能有一个 WebSocket 连接，如果当前已存在一个 WebSocket 连接，会自动关闭该连接，并重新创建一个 WebSocket 连接。**

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| url | String | 是 | 开发者服务器接口地址，必须是 wss 协议，且域名必须是后台配置的合法域名 |
| data | Object | 否 | 请求的数据 |
| header | Object | 否 | HTTP Header , header 中不能设置 Referer |
| method | String | 否 | 默认是GET，有效值为： OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码：**

wx.connectSocket({

 url: 'test.php',

 data:{

 x: '',

 y: ''

 },

 header:{

 'content-type': 'application/json'

 },

 method:"GET"

})

wx.onSocketOpen(CALLBACK)

监听WebSocket连接打开事件。

**示例代码：**

wx.connectSocket({

 url: 'test.php'

})

wx.onSocketOpen(function(res) {

 console.log('WebSocket连接已打开！')

})

wx.onSocketError(CALLBACK)

监听WebSocket错误。

**示例代码：**

wx.connectSocket({

 url: 'test.php'

})

wx.onSocketOpen(function(res){

 console.log('WebSocket连接已打开！')

})

wx.onSocketError(function(res){

 console.log('WebSocket连接打开失败，请检查！')

})

wx.sendSocketMessage(OBJECT)

通过 WebSocket 连接发送数据，需要先 [wx.connectSocket](https://mp.weixin.qq.com/debug/wxadoc/dev/api/network-socket.html#wxconnectsocketobject)，并在 [wx.onSocketOpen](https://mp.weixin.qq.com/debug/wxadoc/dev/api/network-socket.html#wxonsocketopencallback) 回调之后才能发送。

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| data | String/ArrayBuffer | 是 | 需要发送的内容 |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码：**

var socketOpen = false

var socketMsgQueue = []

wx.connectSocket({

 url: 'test.php'

})

wx.onSocketOpen(function(res) {

 socketOpen = true

 for (var i = 0; i < socketMsgQueue.length; i++){

 sendSocketMessage(socketMsgQueue[i])

 }

 socketMsgQueue = []

})

function sendSocketMessage(msg) {

 if (socketOpen) {

 wx.sendSocketMessage({

 data:msg

 })

 } else {

 socketMsgQueue.push(msg)

 }

}

wx.onSocketMessage(CALLBACK)

监听WebSocket接受到服务器的消息事件。

**CALLBACK返回参数：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| data | String/ArrayBuffer | 服务器返回的消息 |

**示例代码：**

wx.connectSocket({

 url: 'test.php'

})

wx.onSocketMessage(function(res) {

 console.log('收到服务器内容：' + res.data)

})

wx.closeSocket()

关闭WebSocket连接。

wx.onSocketClose(CALLBACK)

监听WebSocket关闭。

wx.connectSocket({

 url: 'test.php'

})

//注意这里有时序问题，

//如果 wx.connectSocket 还没回调 wx.onSocketOpen，而先调用 wx.closeSocket，那么就做不到关闭 WebSocket 的目的。

//必须在 WebSocket 打开期间调用 wx.closeSocket 才能关闭。

wx.onSocketOpen(function() {

 wx.closeSocket()

})

wx.onSocketClose(function(res) {

 console.log('WebSocket 已关闭！')

})

Bug & Tip

1. tip: createSocket 链接默认和最大超时时间都是 60s
2. tip: 网络请求的 referer 是不可以设置的，格式固定为 https://servicewechat.com/{appid}/{version}/page-frame.html，其中 {appid} 为小程序的 appid，{version} 为小程序的版本号，版本号为 0 表示为开发版。

### wx.chooseImage(OBJECT)

从本地相册选择图片或使用相机拍照。

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| count | Number | 否 | 最多可以选择的图片张数，默认9 |
| sizeType | StringArray | 否 | original 原图，compressed 压缩图，默认二者都有 |
| sourceType | StringArray | 否 | album 从相册选图，camera 使用相机，默认二者都有 |
| success | Function | 是 | 成功则返回图片的本地文件路径列表 tempFilePaths |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**注：文件的临时路径，在小程序本次启动期间可以正常使用，如需持久保存，需在主动调用**[**wx.saveFile**](https://mp.weixin.qq.com/debug/wxadoc/dev/api/file.html)**，在小程序下次启动时才能访问得到。**

**success返回参数说明：**

| 参数 | 说明 |
| --- | --- |
| tempFilePaths | 图片的本地文件路径列表 |

**示例代码：**

wx.chooseImage({

 count: 1, // 默认9

 sizeType: ['original', 'compressed'], // 可以指定是原图还是压缩图，默认二者都有

 sourceType: ['album', 'camera'], // 可以指定来源是相册还是相机，默认二者都有

 success: function (res) {

 // 返回选定照片的本地文件路径列表，tempFilePath可以作为img标签的src属性显示图片

 var tempFilePaths = res.tempFilePaths

 }

})

### wx.previewImage(OBJECT)

预览图片。

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| current | String | 否 | 当前显示图片的链接，不填则默认为 urls 的第一张 |
| urls | StringArray | 是 | 需要预览的图片链接列表 |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码：**

wx.previewImage({

 current: '', // 当前显示图片的http链接

 urls: [] // 需要预览的图片http链接列表

})

### wx.getImageInfo(OBJECT)

获取图片信息

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| src | String | 是 | 图片的路径，可以是相对路径，临时文件路径，存储文件路径，网络图片路径 |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数说明：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| width | Number | 图片宽度，单位px |
| height | Number | 图片高度 单位px |
| path | String | 返回图片的本地路径 |

**示例代码：**

wx.getImageInfo({

 src: 'images/a.jpg',

 success: function (res) {

 console.log(res.width)

 console.log(res.height)

 }

})

wx.chooseImage({

 success: function (res) {

 wx.getImageInfo({

 src: res.tempFilePaths[0],

 success: function (res) {

 console.log(res.width)

 console.log(res.height)

 }

 })

 }

})

wx.startRecord(OBJECT)

开始录音。当主动调用wx.stopRecord，或者录音超过1分钟时自动结束录音，返回录音文件的临时文件路径。当用户离开小程序时，此接口无法调用。

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| success | Function | 否 | 录音成功后调用，返回录音文件的临时文件路径，res = {tempFilePath: '录音文件的临时路径'} |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**注：文件的临时路径，在小程序本次启动期间可以正常使用，如需持久保存，需在主动调用**[**wx.saveFile**](https://mp.weixin.qq.com/debug/wxadoc/dev/api/file.html)**，在小程序下次启动时才能访问得到。**

**success返回参数说明：**

| 参数 | 说明 |
| --- | --- |
| tempFilePath | 录音文件的临时路径 |

wx.stopRecord()

​ 主动调用停止录音。

**示例代码：**

wx.startRecord({

 success: function(res) {

 var tempFilePath = res.tempFilePath

 },

 fail: function(res) {

 //录音失败

 }

})

setTimeout(function() {

 //结束录音

 wx.stopRecord()

}, 10000)

Bug & Tip

1. tip: wx.startRecord 接口需要用户授权，请兼容用户拒绝授权的场景。

### wx.playVoice(OBJECT)

开始播放语音，同时只允许一个语音文件正在播放，如果前一个语音文件还没播放完，将中断前一个语音播放。

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| filePath | String | 是 | 需要播放的语音文件的文件路径 |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码：**

wx.startRecord({

 success: function(res) {

 var tempFilePath = res.tempFilePath

 wx.playVoice({

 filePath: tempFilePath,

 complete: function(){

 }

 })

 }

})

### wx.pauseVoice()

暂停正在播放的语音。再次调用wx.playVoice播放同一个文件时，会从暂停处开始播放。如果想从头开始播放，需要先调用 wx.stopVoice。

**示例代码：**

wx.startRecord({

 success: function(res) {

 var tempFilePath = res.tempFilePath

 wx.playVoice({

 filePath: tempFilePath

 })

 setTimeout(function() {

 //暂停播放

 wx.pauseVoice()

 }, 5000)

 }

})

### wx.stopVoice()

结束播放语音。

**示例代码：**

wx.startRecord({

 success: function(res) {

 var tempFilePath = res.tempFilePath

 wx.playVoice({

 filePath:tempFilePath

 })

 setTimeout(function(){

 wx.stopVoice()

 }, 5000)

 }

})

wx.getBackgroundAudioPlayerState(OBJECT)

获取后台音乐播放状态。

**OBJECT参数说明：**

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

**success返回参数说明：**

| 参数 | 说明 |
| --- | --- |
| duration | 选定音频的长度（单位：s），只有在当前有音乐播放时返回 |
| currentPosition | 选定音频的播放位置（单位：s），只有在当前有音乐播放时返回 |
| status | 播放状态（2：没有音乐在播放，1：播放中，0：暂停中） |
| downloadPercent | 音频的下载进度（整数，80 代表 80%），只有在当前有音乐播放时返回 |
| dataUrl | 歌曲数据链接，只有在当前有音乐播放时返回 |

**示例代码：**

wx.getBackgroundAudioPlayerState({

 success: function(res) {

 var status = res.status

 var dataUrl = res.dataUrl

 var currentPosition = res.currentPosition

 var duration = res.duration

 var downloadPercent = res.downloadPercent

 }

})

wx.playBackgroundAudio(OBJECT)

使用后台播放器播放音乐，对于微信客户端来说，只能同时有一个后台音乐在播放。当用户离开小程序后，音乐将暂停播放；当用户点击“显示在聊天顶部”时，音乐不会暂停播放；当用户在其他小程序占用了音乐播放器，原有小程序内的音乐将停止播放。

**OBJECT参数说明**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| dataUrl | String | 是 | 音乐链接 |
| title | String | 否 | 音乐标题 |
| coverImgUrl | String | 否 | 封面URL |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码**

wx.playBackgroundAudio({

 dataUrl: '',

 title: '',

 coverImgUrl: ''

})

wx.pauseBackgroundAudio()

暂停播放音乐。

**示例代码**

wx.pauseBackgroundAudio()

wx.seekBackgroundAudio(OBJECT)

控制音乐播放进度。

**OBJECT参数说明**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| position | Number | 是 | 音乐位置，单位：秒 |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码**

wx.seekBackgroundAudio({

 position: 30

})

wx.stopBackgroundAudio()

停止播放音乐。

**示例代码**

wx.stopBackgroundAudio()

wx.onBackgroundAudioPlay(CALLBACK)

监听音乐播放。

wx.onBackgroundAudioPause(CALLBACK)

监听音乐暂停。

wx.onBackgroundAudioStop(CALLBACK)

监听音乐停止。

bug & tip

1. bug: iOS 6.3.30 wx.seekBackgroundAudio 会有短暂延迟

### wx.createAudioContext(audioId)

创建并返回 audio 上下文 audioContext 对象

#### audioContext

audioContext 通过 audioId 跟一个 <audio/> 组件绑定，通过它可以操作对应的 <audio/> 组件。

**audioContext 对象的方法列表：**

| 方法 | 参数 | 说明 |
| --- | --- | --- |
| setSrc | src | 音频的地址 |
| play | 无 | 播放 |
| pause | 无 | 暂停 |
| seek | position | 跳转到指定位置，单位 s |

**示例代码：**

<!-- audio.wxml -->

<audio src="{{src}}" id="myAudio" ></audio>

<button type="primary" bindtap="audioPlay">播放</button>

<button type="primary" bindtap="audioPause">暂停</button>

<button type="primary" bindtap="audio14">设置当前播放时间为14秒</button>

<button type="primary" bindtap="audioStart">回到开头</button>

// audio.js

Page({

 onReady: function (e) {

 // 使用 wx.createAudioContext 获取 audio 上下文 context

 this.audioCtx = wx.createAudioContext('myAudio')

 this.audioCtx.setSrc('http://ws.stream.qqmusic.qq.com/M500001VfvsJ21xFqb.mp3?guid=ffffffff82def4af4b12b3cd9337d5e7&uin=346897220&vkey=6292F51E1E384E06DCBDC9AB7C49FD713D632D313AC4858BACB8DDD29067D3C601481D36E62053BF8DFEAF74C0A5CCFADD6471160CAF3E6A&fromtag=46')

 this.audioCtx.play()

 },

 data: {

 src: ''

 },

 audioPlay: function () {

 this.audioCtx.play()

 },

 audioPause: function () {

 this.audioCtx.pause()

 },

 audio14: function () {

 this.audioCtx.seek(14)

 },

 audioStart: function () {

 this.audioCtx.seek(0)

 }

})

### wx.chooseVideo(OBJECT)

拍摄视频或从手机相册中选视频，返回视频的临时文件路径。

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| sourceType | StringArray | 否 | album 从相册选视频，camera 使用相机拍摄，默认为：['album', 'camera'] |
| maxDuration | Number | 否 | 拍摄视频最长拍摄时间，单位秒。最长支持 60 秒 |
| camera | String | 否 | 默认调起的为前置还是后置摄像头。front: 前置，back: 后置，默认 back |
| success | Function | 否 | 接口调用成功，返回视频文件的临时文件路径，详见返回参数说明 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数说明：**

| 参数 | 说明 |
| --- | --- |
| tempFilePath | 选定视频的临时文件路径 |
| duration | 选定视频的时间长度 |
| size | 选定视频的数据量大小 |
| height | 返回选定视频的长 |
| width | 返回选定视频的宽 |

**注：文件的临时路径，在小程序本次启动期间可以正常使用，如需持久保存，需在主动调用**[**wx.saveFile**](https://mp.weixin.qq.com/debug/wxadoc/dev/api/file.html)**，在小程序下次启动时才能访问得到。**

**示例代码：**

<view class="container">

 <video src="{{src}}"></video>

 <button bindtap="bindButtonTap">获取视频</button>

</view>

Page({

 bindButtonTap: function() {

 var that = this

 wx.chooseVideo({

 sourceType: ['album','camera'],

 maxDuration: 60,

 camera: 'back',

 success: function(res) {

 that.setData({

 src: res.tempFilePath

 })

 }

 })

 }

})

### wx.createVideoContext(videoId)

创建并返回 video 上下文 videoContext 对象

#### videoContext

videoContext 通过 videoId 跟一个 video 组件绑定，通过它可以操作一个 video 组件。

**videoContext 对象的方法列表：**

| 方法 | 参数 | 说明 |
| --- | --- | --- |
| play | 无 | 播放 |
| pause | 无 | 暂停 |
| seek | position | 跳转到指定位置，单位 s |
| sendDanmu | danmu | 发送弹幕，danmu 包含两个属性 text, color。 |

**示例代码：**

<view class="section tc">

 <video id="myVideo" src="http://wxsnsdy.tc.qq.com/105/20210/snsdyvideodownload?filekey=30280201010421301f0201690402534804102ca905ce620b1241b726bc41dcff44e00204012882540400&bizid=1023&hy=SH&fileparam=302c020101042530230204136ffd93020457e3c4ff02024ef202031e8d7f02030f42400204045a320a0201000400" enable-danmu danmu-btn controls></video>

 <view class="btn-area">

 <input bindblur="bindInputBlur"/>

 <button bindtap="bindSendDanmu">发送弹幕</button>

 </view>

</view>

function getRandomColor () {

 let rgb = []

 for (let i = 0 ; i < 3; ++i){

 let color = Math.floor(Math.random() \* 256).toString(16)

 color = color.length == 1 ? '0' + color : color

 rgb.push(color)

 }

 return '#' + rgb.join('')

}

Page({

 onReady: function (res) {

 this.videoContext = wx.createVideoContext('myVideo')

 },

 inputValue: '',

 bindInputBlur: function(e) {

 this.inputValue = e.detail.value

 },

 bindSendDanmu: function () {

 this.videoContext.sendDanmu({

 text: this.inputValue,

 color: getRandomColor()

 })

 }

})

wx.saveFile(OBJECT)

保存文件到本地。

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| tempFilePath | String | 是 | 需要保存的文件的临时路径 |
| success | Function | 否 | 返回文件的保存路径，res = {savedFilePath: '文件的保存路径'} |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数说明：**

| 参数 | 说明 |
| --- | --- |
| savedFilePath | 文件的保存路径 |

**示例代码：**

wx.chooseImage({

 success: function(res) {

 var tempFilePaths = res.tempFilePaths

 wx.saveFile({

 tempFilePath: tempFilePaths[0],

 success: function(res) {

 var savedFilePath = res.savedFilePath

 }

 })

 }

})

bug & tip

1. tip: 本地文件存储的大小限制为 10M

wx.getSavedFileList(OBJECT)

获取本地已保存的文件列表

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| success | Function | 否 | 接口调用成功的回调函数，返回结果见success返回参数说明 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数说明：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| errMsg | String | 接口调用结果 |
| fileList | Object Array | 文件列表 |

**fileList中的项目说明：**

| 键 | 类型 | 说明 |
| --- | --- | --- |
| filePath | String | 文件的本地路径 |
| createTime | Number | 文件的保存时的时间戳，从1970/01/01 08:00:00 到当前时间的秒数 |
| size | Number | 文件大小，单位B |

**示例代码：**

wx.getSavedFileList({

 success: function(res) {

 console.log(res.fileList)

 }

})

wx.getSavedFileInfo(OBJECT)

获取本地文件的文件信息

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| filePath | String | 是 | 文件路径 |
| success | Function | 否 | 接口调用成功的回调函数，返回结果见success返回参数说明 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数说明：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| errMsg | String | 接口调用结果 |
| size | Number | 文件大小，单位B |
| createTime | Number | 文件的保存是的时间戳，从1970/01/01 08:00:00 到当前时间的秒数 |

**示例代码：**

wx.getSavedFileInfo({

 filePath: 'wxfile://somefile', //仅做示例用，非真正的文件路径

 success: function(res) {

 console.log(res.size)

 console.log(res.createTime)

 }

})

wx.removeSavedFile(OBJECT)

删除本地存储的文件

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| filePath | String | 是 | 需要删除的文件路径 |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码：**

wx.getSavedFileList({

 success: function(res) {

 if (res.fileList.length > 0){

 wx.removeSavedFile({

 filePath: res.fileList[0].filePath,

 complete: function(res) {

 console.log(res)

 }

 })

 }

 }

})

wx.openDocument(OBJECT)

新开页面打开文档，支持格式：doc, xls, ppt, pdf, docx, xlsx, pptx

**OBJECT参数说明：**

| 参数 | 说明 | 必填 | 说明 |
| --- | --- | --- | --- |
| filePath | String | 是 | 文件路径，可通过 downFile 获得 |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

示例代码

wx.downloadFile({

 url: 'http://example.com/somefile.pdf',

 success: function (res) {

 var filePath = res.tempFilePath

 wx.openDocument({

 filePath: filePath,

 success: function (res) {

 console.log('打开文档成功')

 }

 })

 }

})

wx.setStorage(OBJECT)

将数据存储在本地缓存中指定的 key 中，会覆盖掉原来该 key 对应的内容，这是一个异步接口。

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| key | String | 是 | 本地缓存中的指定的 key |
| data | Object/String | 是 | 需要存储的内容 |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码：**

wx.setStorage({

 key:"key",

 data:"value"

})

wx.setStorageSync(KEY,DATA)

将 data 存储在本地缓存中指定的 key 中，会覆盖掉原来该 key 对应的内容，这是一个同步接口。

**参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| key | String | 是 | 本地缓存中的指定的 key |
| data | Object/String | 是 | 需要存储的内容 |

**示例代码**

try {

 wx.setStorageSync('key', 'value')

} catch (e) {

}

wx.getStorage(OBJECT)

从本地缓存中异步获取指定 key 对应的内容。

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| key | String | 是 | 本地缓存中的指定的 key |
| success | Function | 是 | 接口调用的回调函数,res = {data: key对应的内容} |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数说明：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| data | String | key对应的内容 |

**示例代码：**

wx.getStorage({

 key: 'key',

 success: function(res) {

 console.log(res.data)

 }

})

wx.getStorageSync(KEY)

从本地缓存中同步获取指定 key 对应的内容。

**参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| key | String | 是 | 本地缓存中的指定的 key |

**示例代码：**

try {

 var value = wx.getStorageSync('key')

 if (value) {

 // Do something with return value

 }

} catch (e) {

 // Do something when catch error

}

wx.getStorageInfo(OBJECT)

异步获取当前storage的相关信息

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| success | Function | 是 | 接口调用的回调函数，详见返回参数说明 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数说明：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| keys | String Array | 当前storage中所有的key |
| currentSize | Number | 当前占用的空间大小, 单位kb |
| limitSize | Number | 限制的空间大小，单位kb |

**示例代码：**

wx.getStorageInfo({

 success: function(res) {

 console.log(res.keys)

 console.log(res.currentSize)

 console.log(res.limitSize)

 }

})

wx.getStorageInfoSync

同步获取当前storage的相关信息

**示例代码：**

try {

 var res = wx.getStorageInfoSync()

 console.log(res.keys)

 console.log(res.currentSize)

 console.log(res.limitSize)

} catch (e) {

 // Do something when catch error

}

wx.removeStorage(OBJECT)

从本地缓存中异步移除指定 key 。

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| key | String | 是 | 本地缓存中的指定的 key |
| success | Function | 是 | 接口调用的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码：**

wx.removeStorage({

 key: 'key',

 success: function(res) {

 console.log(res.data)

 }

})

wx.removeStorageSync(KEY)

从本地缓存中同步移除指定 key 。

**参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| key | String | 是 | 本地缓存中的指定的 key |

**示例代码：**

try {

 wx.removeStorageSync('key')

} catch (e) {

 // Do something when catch error

}

wx.clearStorage()

清理本地数据缓存。

**示例代码：**

wx.clearStorage()

wx.clearStorageSync()

同步清理本地数据缓存

**示例代码：**

try {

 wx.clearStorageSync()

} catch(e) {

 // Do something when catch error

}

Bug & Tip

1. tip: 本地数据存储的大小限制为 10MB

### wx.getLocation(OBJECT)

获取当前的地理位置、速度。当用户离开小程序后，此接口无法调用；当用户点击“显示在聊天顶部”时，此接口可继续调用。

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| type | String | 否 | 默认为 wgs84 返回 gps 坐标，gcj02 返回可用于wx.openLocation的坐标 |
| success | Function | 是 | 接口调用成功的回调函数，返回内容详见返回参数说明。 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数说明：**

| 参数 | 说明 |
| --- | --- |
| latitude | 纬度，浮点数，范围为-90~90，负数表示南纬 |
| longitude | 经度，浮点数，范围为-180~180，负数表示西经 |
| speed | 速度，浮点数，单位m/s |
| accuracy | 位置的精确度 |

**示例代码：**

wx.getLocation({

 type: 'wgs84',

 success: function(res) {

 var latitude = res.latitude

 var longitude = res.longitude

 var speed = res.speed

 var accuracy = res.accuracy

 }

})

### wx.chooseLocation(OBJECT)

打开地图选择位置

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| success | Function | 是 | 接口调用成功的回调函数，返回内容详见返回参数说明。 |
| cancel | Function | 否 | 用户取消时调用 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数说明：**

| 参数 | 说明 |
| --- | --- |
| name | 位置名称 |
| address | 详细地址 |
| latitude | 纬度，浮点数，范围为-90~90，负数表示南纬 |
| longitude | 经度，浮点数，范围为-180~180，负数表示西经 |

### wx.openLocation(OBJECT)

​ 使用微信内置地图查看位置

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| latitude | Float | 是 | 纬度，范围为-90~90，负数表示南纬 |
| longitude | Float | 是 | 经度，范围为-180~180，负数表示西经 |
| scale | INT | 否 | 缩放比例，范围5~18，默认为18 |
| name | String | 否 | 位置名 |
| address | String | 否 | 地址的详细说明 |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码：**

wx.getLocation({

 type: 'gcj02', //返回可以用于wx.openLocation的经纬度

 success: function(res) {

 var latitude = res.latitude

 var longitude = res.longitude

 wx.openLocation({

 latitude: latitude,

 longitude: longitude,

 scale: 28

 })

 }

})

#### Bug & Tip

1. bug: iOS 6.3.30 type 参数不生效，只会返回 wgs84 类型的坐标信息
2. tip: wx.getLocation、wx.chooseLocation 接口需要用户授权，请兼容用户拒绝授权的场景。

wx.createMapContext(mapId)

创建并返回 map 上下文 mapContext 对象

mapContext

mapContext 通过 mapId 跟一个 <map/> 组件绑定，通过它可以操作对应的 <map/> 组件。

**mapContext 对象的方法列表**

| 方法 | 参数 | 说明 |
| --- | --- | --- |
| getCenterLocation | OBJECT | 获取当前地图中心的经纬度，返回的是 gcj02 坐标系，可以用于 [wx.openLocation](https://mp.weixin.qq.com/debug/wxadoc/dev/api/location.html#wxopenlocationobject) |
| moveToLocation | 无 | 将地图中心移动到当前定位点，需要配合map组件的show-location使用 |

**getCenterLocation 的 OBJECT 参数列表**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| success | Function | 否 | 接口调用成功的回调函数 ，res = { longitude: "经度", latitude: "纬度"} |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码：**

<!-- map.wxml -->

<map id="myMap" show-location />

<button type="primary" bindtap="getCenterLocation">获取位置</button>

<button type="primary" bindtap="moveToLocation">移动位置</button>

// map.js

Page({

 onReady: function (e) {

 // 使用 wx.createMapContext 获取 map 上下文

 this.mapCtx = wx.createMapContext('myMap')

 },

 getCenterLocation: function () {

 this.mapCtx.getCenterLocation({

 success: function(res){

 console.log(res.longitude)

 console.log(res.latitude)

 }

 })

 },

 moveToLocation: function () {

 this.mapCtx.moveToLocation()

 }

})

### wx.getSystemInfo(OBJECT)

获取系统信息。

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| success | Function | 是 | 接口调用成功的回调 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success回调参数说明：**

| 参数 | 说明 | 最低版本 |
| --- | --- | --- |
| model | 手机型号 |  |
| pixelRatio | 设备像素比 |  |
| screenWidth | 屏幕宽度 | [1.1.0](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html) |
| screenHeight | 屏幕高度 | [1.1.0](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html) |
| windowWidth | 可使用窗口宽度 |  |
| windowHeight | 可使用窗口高度 |  |
| language | 微信设置的语言 |  |
| version | 微信版本号 |  |
| system | 操作系统版本 |  |
| platform | 客户端平台 |  |
| SDKVersion | 客户端基础库版本 | [1.1.0](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html) |

**示例代码：**

wx.getSystemInfo({

 success: function(res) {

 console.log(res.model)

 console.log(res.pixelRatio)

 console.log(res.windowWidth)

 console.log(res.windowHeight)

 console.log(res.language)

 console.log(res.version)

 console.log(res.platform)

 }

})

### wx.getSystemInfoSync()

获取系统信息同步接口

**同步返回参数说明：**

| 参数 | 说明 | 最低版本 |
| --- | --- | --- |
| model | 手机型号 |  |
| pixelRatio | 设备像素比 |  |
| screenWidth | 屏幕宽度 | [1.1.0](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html) |
| screenHeight | 屏幕高度 | [1.1.0](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html) |
| windowWidth | 可使用窗口宽度 |  |
| windowHeight | 可使用窗口高度 |  |
| language | 微信设置的语言 |  |
| version | 微信版本号 |  |
| system | 操作系统版本 |  |
| platform | 客户端平台 |  |
| SDKVersion | 客户端基础库版本 | [1.1.0](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html) |

**示例代码：**

try {

 var res = wx.getSystemInfoSync()

 console.log(res.model)

 console.log(res.pixelRatio)

 console.log(res.windowWidth)

 console.log(res.windowHeight)

 console.log(res.language)

 console.log(res.version)

 console.log(res.platform)

} catch (e) {

 // Do something when catch error

}

### wx.canIUse(String)

判断小程序的API，回调，参数，组件等是否在当前版本可用。

**String参数说明：** 使用${API}.${method}.${param}.${options}或者${component}.${attribute}.${option}方式来调用，例如：

* ${API} 代表 API 名字
* ${method} 代表调用方式，有效值为return, success, object, callback
* ${param} 代表参数或者返回值
* ${options} 代表参数的可选值
* ${component} 代表组件名字
* ${attribute} 代表组件属性
* ${option} 代表组件属性的可选值

例子：

wx.canIUse('openBluetoothAdapter')

wx.canIUse('getSystemInfoSync.return.screenWidth')

wx.canIUse('getSystemInfo.success.screenWidth')

wx.canIUse('showToast.object.image')

wx.canIUse('onCompassChange.callback.direction')

wx.canIUse('request.object.method.GET')

wx.canIUse('contact-button')

wx.canIUse('text.selectable')

wx.canIUse('button.open-type.contact')

### wx.getNetworkType(OBJECT)

获取网络类型。

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| success | Function | 是 | 接口调用成功，返回网络类型 networkType |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数说明：**

| 参数 | 说明 |
| --- | --- |
| networkType | 网络类型 |

wx.getNetworkType({

 success: function(res) {

 // 返回网络类型, 有效值：

 // wifi/2g/3g/4g/unknown(Android下不常见的网络类型)/none(无网络)

 var networkType = res.networkType

 }

})

### wx.onNetworkStatusChange(CALLBACK)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

监听网络状态变化。

**CALLBACK返回参数：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| isConnected | Boolean | 当前是否有网络连接 |
| networkType | String | 网络类型 |

**networkType 有效值：**

| 值 | 说明 |
| --- | --- |
| wifi | wifi 网络 |
| 2g | 2g 网络 |
| 3g | 3g 网络 |
| 4g | 4g 网络 |
| none | 无网络 |
| unknown | Android下不常见的网络类型 |

**示例代码：**

wx.onNetworkStatusChange(function(res) {

 console.log(res.isConnected)

 console.log(res.networkType)

})

### wx.onAccelerometerChange(CALLBACK)

监听加速度数据，频率：5次/秒，接口调用后会自动开始监听，可使用 wx.stopAccelerometer 停止监听。

**CALLBACK返回参数：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| x | Number | X 轴 |
| y | Number | Y 轴 |
| z | Number | Z 轴 |

**示例代码：**

wx.onAccelerometerChange(function(res) {

 console.log(res.x)

 console.log(res.y)

 console.log(res.z)

})

### wx.startAccelerometer(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

开始监听加速度数据。

**OBJECT参数说明：**

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

**示例代码：**

wx.startAccelerometer()

### wx.stopAccelerometer(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

停止监听加速度数据。

**OBJECT参数说明：**

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

**示例代码：**

wx.stopAccelerometer()

### wx.onCompassChange(CALLBACK)

监听罗盘数据，频率：5次/秒，接口调用后会自动开始监听，可使用wx.stopCompass停止监听。

**CALLBACK返回参数：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| direction | Number | 面对的方向度数 |

**示例代码：**

wx.onCompassChange(function (res) {

 console.log(res.direction)

})

### wx.startCompass(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

开始监听罗盘数据。

**OBJECT参数说明：**

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

**示例代码：**

wx.startCompass()

### wx.stopCompass(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

停止监听罗盘数据。

**OBJECT参数说明：**

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

**示例代码：**

wx.stopCompass()

wx.makePhoneCall(OBJECT)

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| phoneNumber | String | 是 | 需要拨打的电话号码 |
| success | Function | 否 | 接口调用成功的回调 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码：**

wx.makePhoneCall({

 phoneNumber: '1340000' //仅为示例，并非真实的电话号码

})

wx.scanCode(OBJECT)

调起客户端扫码界面，扫码成功后返回对应的结果

**Object 参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| success | Function | 否 | 接口调用成功的回调函数，返回内容详见返回参数说明。 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数说明：**

| 参数 | 说明 |
| --- | --- |
| result | 所扫码的内容 |
| scanType | 所扫码的类型 |
| charSet | 所扫码的字符集 |
| path | 当所扫的码为当前小程序的合法二维码时，会返回此字段，内容为二维码携带的 path |

**示例代码：**

wx.scanCode({

 success: (res) => {

 console.log(res)

 }

})

### wx.setClipboardData(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

设置系统剪贴板的内容

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| data | String | 是 | 需要设置的内容 |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码：**

wx.setClipboardData({

 data: 'data',

 success: function(res) {

 wx.getClipboardData({

 success: function(res) {

 console.log(res.data) // data

 }

 })

 }

})

### wx.getClipboardData(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

获取系统剪贴板内容

**OBJECT参数说明：**

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

**success返回参数说明：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| data | String | 剪贴板的内容 |

**示例代码：**

wx.getClipboardData({

 success: function(res){

 console.log(res.data)

 }

})

### wx.openBluetoothAdapter(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

初始化蓝牙适配器

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| success | Function | 是 | 成功则返回成功初始化信息 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码：**

wx.openBluetoothAdapter({

 success: function (res) {

 console.log(res)

 }

})

#### Bug & Tip

1. tip: 由于系统的问题，目前仅在 mac 版的开发工具上支持蓝牙调试
2. tip: 基础库版本 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

### wx.closeBluetoothAdapter(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

关闭蓝牙模块。调用该方法将断开所有已建立的链接并释放系统资源

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| success | Function | 是 | 成功则返回成功关闭模块信息 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码：**

wx.closeBluetoothAdapter({

 success: function (res) {

 console.log(res)

 }

})

### wx.getBluetoothAdapterState(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

获取本机蓝牙适配器状态

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| success | Function | 是 | 成功则返回本机蓝牙适配器状态 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| adapterState | object | 蓝牙适配器信息 |
| errMsg | string | 成功：ok，错误：详细信息 |

#### adapterState

蓝牙适配器状态信息

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| discovering | boolean | 是否正在搜索设备 |
| available | boolean | 蓝牙适配器是否可用 |

**示例代码：**

wx.getBluetoothAdapterState({

 success: function (res) {

 console.log(res)

 }

})

### wx.onBluetoothAdapterStateChange(CALLBACK)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

监听蓝牙适配器状态变化事件

**CALLBACK参数说明：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| available | boolean | 蓝牙适配器是否可用 |
| discovering | boolean | 蓝牙适配器是否处于搜索状态 |

**示例代码：**

wx.onBluetoothAdapterStateChange(function(res) {

 console.log(`adapterState changed, now is`, res)

})

### wx.startBluetoothDevicesDiscovery(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

开始搜寻附近的蓝牙外围设备。注意，该操作比较耗费系统资源，请在搜索并连接到设备后调用 stop 方法停止搜索。

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| services | Array | 否 | 蓝牙设备主 service 的 uuid 列表 |
| success | Function | 是 | 成功则返回本机蓝牙适配器状态 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**services**参数说明：某些蓝牙设备会广播自己的主 service 的 uuid。如果这里传入该数组，那么根据该 uuid 列表，只搜索有这个主服务的设备。

**success返回参数：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| errMsg | string | 成功：ok，错误：详细信息 |

**示例代码：**

// 以微信硬件平台的蓝牙智能灯为例，主服务的 UUID 是 FEE7。传入这个参数，只搜索主服务 UUID 为 FEE7 的设备

wx.startBluetoothDevicesDiscovery({

 services: ['FEE7'],

 success: function (res) {

 console.log(res)

 }

})

### wx.stopBluetoothDevicesDiscovery(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

停止搜寻附近的蓝牙外围设备。请在确保找到需要连接的设备后调用该方法停止搜索。

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| success | Function | 是 | 成功则返回本机蓝牙适配器状态 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| errMsg | string | 成功：ok，错误：详细信息 |

#### adapterState

蓝牙适配器状态信息

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| discovering | boolean | 是否正在搜索设备 |
| available | boolean | 蓝牙适配器是否可用 |

**示例代码：**

wx.stopBluetoothDevicesDiscovery({

 success: function (res) {

 console.log(res)

 }

})

### wx.getBluetoothDevices(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

获取所有已发现的蓝牙设备，包括已经和本机处于连接状态的设备

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| services | Array | 否 | 蓝牙设备主 service 的 uuid 列表 |
| success | Function | 是 | 成功则返回本机蓝牙适配器状态 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| devices | Array | uuid 对应的的已连接设备列表 |
| errMsg | string | 成功：ok，错误：详细信息 |

#### device 对象

蓝牙设备信息

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| name | string | 蓝牙设备名称，某些设备可能没有 |
| deviceId | string | 用于区分设备的 id |
| RSSI | int | 当前蓝牙设备的信号强度 |
| advertisData | ArrayBuffer | 当前蓝牙设备的广播内容 |

**示例代码：**

wx.getBluetoothDevices({

 success: function (res) {

 console.log(res)

 }

})

#### Bug & Tip

1. tip: Mac系统可能无法获取advertisData及RSSI，请使用真机调试
2. tip: 开发者工具和 Android 上获取到的deviceId为设备 MAC 地址，iOS 上则为设备 uuid。因此deviceId不能硬编码到代码中

### wx.onBluetoothDeviceFound(CALLBACK)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

监听寻找到新设备的事件

**CALLBACK参数说明：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| devices | Array | 新搜索到的设备列表 |

#### device对象

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| deviceId | string | 蓝牙设备 id，参考 device 对象 |
| name | string | 蓝牙设备名称，参考 device 对象 |
| RSSI | int | 当前蓝牙设备的信号强度 |
| advertisData | ArrayBuffer | 当前蓝牙设备的广播内容 |

**示例代码：**

wx.onBluetoothDeviceFound(function(devices) {

 console.log('new device list has founded')

 console.dir(devices)

})

#### Bug & Tip

1. tip: Mac系统可能无法获取advertisData及RSSI，请使用真机调试
2. tip: 开发者工具和 Android 上获取到的deviceId为设备 MAC 地址，iOS 上则为设备 uuid。因此deviceId不能硬编码到代码中

### wx.getConnectedBluetoothDevices(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

根据 uuid 获取处于已连接状态的设备

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| services | Array | 是 | 蓝牙设备主 service 的 uuid 列表 |
| success | Function | 是 | 成功则返回本机蓝牙适配器状态 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| devices | Array | 搜索到的设备列表 |
| errMsg | string | 成功：ok，错误：详细信息 |

#### device对象

蓝牙设备信息

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| name | string | 蓝牙设备名称，某些设备可能没有 |
| deviceId | string | 用于区分设备的 id |

**示例代码：**

wx.getConnectedBluetoothDevices({

 success: function (res) {

 console.log(res)

 }

})

#### Bug & Tip

1. tip: 开发者工具和 Android 上获取到的deviceId为设备 MAC 地址，iOS 上则为设备 uuid。因此deviceId不能硬编码到代码中

## 低功耗蓝牙接口

### wx.createBLEConnection(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

连接低功耗蓝牙设备

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| deviceId | string | 是 | 蓝牙设备 id，参考 getDevices 接口 |
| success | Function | 是 | 成功则返回本机蓝牙适配器状态 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| errMsg | string | 成功：ok，错误：详细信息 |

**示例代码：**

wx.createBLEConnection({

 // 这里的 deviceId 需要在上面的 getBluetoothDevices 或 onBluetoothDeviceFound 接口中获取

 deviceId: deviceId,

 success: function (res) {

 console.log(res)

 }

})

### wx.closeBLEConnection(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

断开与低功耗蓝牙设备的连接

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| deviceId | string | 是 | 蓝牙设备 id，参考 getDevices 接口 |
| success | Function | 是 | 成功则返回本机蓝牙适配器状态 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| errMsg | string | 成功：ok，错误：详细信息 |

**示例代码：**

wx.closeBLEConnection({

 success: function (res) {

 console.log(res)

 }

})

### wx.getBLEDeviceServices(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

获取蓝牙设备所有 service（服务）

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| deviceId | string | 是 | 蓝牙设备 id，参考 getDevices 接口 |
| success | Function | 是 | 成功则返回本机蓝牙适配器状态 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| services | array | 设备服务列表 |
| errMsg | string | 成功：ok，错误：详细信息 |

#### service对象

蓝牙设备service(服务)信息

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| uuid | string | 蓝牙设备服务的 uuid |
| isPrimary | boolean | 该服务是否为主服务 |

**示例代码：**

wx.getBLEDeviceServices({

 // 这里的 deviceId 需要在上面的 getBluetoothDevices 或 onBluetoothDeviceFound 接口中获取

 deviceId: deviceId,

 success: function (res) {

 console.log('device services:', res.services)

 }

})

### wx.getBLEDeviceCharacteristics(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

获取蓝牙设备所有 characteristic（特征值）

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| deviceId | string | 是 | 蓝牙设备 id，参考 device 对象 |
| serviceId | string | 是 | 蓝牙服务 uuid |
| success | Function | 是 | 成功则返回本机蓝牙适配器状态 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| characteristics | array | 设备特征值列表 |
| errMsg | string | 成功：ok，错误：详细信息 |

#### characteristic对象

蓝牙设备characteristic(特征值)信息

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| uuid | string | 蓝牙设备特征值的 uuid |
| properties | object | 该特征值支持的操作类型 |

##### properties对象

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| read | boolean | 该特征值是否支持 read 操作 |
| write | boolean | 该特征值是否支持 write 操作 |
| notify | boolean | 该特征值是否支持 notify 操作 |
| indicate | boolean | 该特征值是否支持 indicate 操作 |

**示例代码：**

wx.getBLEDeviceCharacteristics({

 // 这里的 deviceId 需要在上面的 getBluetoothDevices 或 onBluetoothDeviceFound 接口中获取

 deviceId: deviceId,

 // 这里的 serviceId 需要在上面的 getBLEDeviceServices 接口中获取

 serviceId: serviceId,

 success: function (res) {

 console.log('device getBLEDeviceCharacteristics:', res.characteristics)

 }

})

### wx.readBLECharacteristicValue(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

读取低功耗蓝牙设备的特征值的二进制数据值。注意：必须设备的特征值支持read才可以成功调用，具体参照 characteristic 的 properties 属性

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| deviceId | string | 是 | 蓝牙设备 id，参考 device 对象 |
| serviceId | string | 是 | 蓝牙特征值对应服务的 uuid |
| characteristicId | string | 是 | 蓝牙特征值的 uuid |
| success | Function | 是 | 成功则返回本机蓝牙适配器状态 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| characteristic | object | 设备特征值信息 |
| errMsg | string | 成功：ok，错误：详细信息 |

#### characteristic对象

蓝牙设备characteristic(特征值)信息

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| characteristicId | string | 蓝牙设备特征值的 uuid |
| serviceId | object | 蓝牙设备特征值对应服务的 uuid |
| value | ArrayBuffer | 蓝牙设备特征值对应的二进制值 |

**示例代码：**

// 必须在这里的回调才能获取

wx.onBLECharacteristicValueChange(function(characteristic) {

 console.log('characteristic value comed:', characteristic)

})

wx.readBLECharacteristicValue({

 // 这里的 deviceId 需要在上面的 getBluetoothDevices 或 onBluetoothDeviceFound 接口中获取

 deviceId: deviceId,

 // 这里的 serviceId 需要在上面的 getBLEDeviceServices 接口中获取

 serviceId: serviceId,

 // 这里的 characteristicId 需要在上面的 getBLEDeviceCharacteristics 接口中获取

 characteristicId: characteristicId,

 success: function (res) {

 console.log('readBLECharacteristicValue:', res.characteristic.value)

 }

})

#### Bug & Tip

1. tip: 并行调用多次读写接口存在读写失败的可能性。
2. tip: read接口读取到的信息需要在onBLECharacteristicValueChange方法注册的回调中获取。

### wx.writeBLECharacteristicValue(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

向低功耗蓝牙设备特征值中写入二进制数据。注意：必须设备的特征值支持write才可以成功调用，具体参照 characteristic 的 properties 属性

tips: 并行调用多次读写接口存在读写失败的可能性

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| deviceId | string | 是 | 蓝牙设备 id，参考 device 对象 |
| serviceId | string | 是 | 蓝牙特征值对应服务的 uuid |
| characteristicId | string | 是 | 蓝牙特征值的 uuid |
| value | ArrayBuffer | 是 | 蓝牙设备特征值对应的二进制值 |
| success | Function | 是 | 成功则返回本机蓝牙适配器状态 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| errMsg | string | 成功：ok，错误：详细信息 |

**示例代码：**

// 这里的回调可以获取到 write 导致的特征值改变

wx.onBLECharacteristicValueChange(function(characteristic) {

 console.log('characteristic value changed:', characteristic)

})

// 向蓝牙设备发送一个0x00的16进制数据

let buffer = new ArrayBuffer(1)

let dataView = new DataView(buffer)

dataView.setUint8(0, 0)

wx.writeBLECharacteristicValue({

 // 这里的 deviceId 需要在上面的 getBluetoothDevices 或 onBluetoothDeviceFound 接口中获取

 deviceId: deviceId,

 // 这里的 serviceId 需要在上面的 getBLEDeviceServices 接口中获取

 serviceId: serviceId,

 // 这里的 characteristicId 需要在上面的 getBLEDeviceCharacteristics 接口中获取

 characteristicId: characteristicId,

 // 这里的value是ArrayBuffer类型

 value: buffer,

 success: function (res) {

 console.log('writeBLECharacteristicValue success', res.errMsg)

 }

})

### wx.notifyBLECharacteristicValueChange(OBJECT)

基础库 1.1.1 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

启用低功耗蓝牙设备特征值变化时的 notify 功能。注意：必须设备的特征值支持notify才可以成功调用，具体参照 characteristic 的 properties 属性

另外，必须先启用notify才能监听到设备 characteristicValueChange 事件

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| deviceId | string | 是 | 蓝牙设备 id，参考 device 对象 |
| serviceId | string | 是 | 蓝牙特征值对应服务的 uuid |
| characteristicId | string | 是 | 蓝牙特征值的 uuid |
| state | boolean | 是 | true: 启用 notify; false: 停用 notify |
| success | Function | 是 | 成功则返回本机蓝牙适配器状态 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| errMsg | string | 成功：ok，错误：详细信息 |

**示例代码：**

wx.notifyBLECharacteristicValueChange({

 state: true, // 启用 notify 功能

 // 这里的 deviceId 需要在上面的 getBluetoothDevices 或 onBluetoothDeviceFound 接口中获取

 deviceId: deviceId,

 // 这里的 serviceId 需要在上面的 getBLEDeviceServices 接口中获取

 serviceId: serviceId,

 // 这里的 characteristicId 需要在上面的 getBLEDeviceCharacteristics 接口中获取

 characteristicId: characteristicId,

 success: function (res) {

 console.log('notifyBLECharacteristicValueChange success', res.errMsg)

 }

})

### wx.onBLEConnectionStateChange(CALLBACK)

基础库 1.1.1 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

监听低功耗蓝牙连接的错误事件，包括设备丢失，连接异常断开等等。

**CALLBACK参数说明：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| deviceId | string | 蓝牙设备 id，参考 device 对象 |
| connected | boolean | 连接目前的状态 |

**示例代码：**

wx.onBLEConnectionStateChange(function(res) {

 // 该方法回调中可以用于处理连接意外断开等异常情况

 console.log(`device ${res.deviceId} state has changed, connected: ${res.connected}`)

})

### wx.onBLECharacteristicValueChange(CALLBACK)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

监听低功耗蓝牙设备的特征值变化。必须先启用notify接口才能接收到设备推送的notification。

**CALLBACK参数说明：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| deviceId | string | 蓝牙设备 id，参考 device 对象 |
| serviceId | string | 特征值所属服务 uuid |
| characteristicId | string | 特征值 uuid |
| value | ArrayBuffer | 特征值最新的值 |

**示例代码：**

wx.onBLECharacteristicValueChange(function(res) {

 console.log(`characteristic ${res.characteristicId} has changed, now is ${res.value}`)

})

### wx.showToast(OBJECT)

显示消息提示框

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 | 最低版本 |
| --- | --- | --- | --- | --- |
| title | String | 是 | 提示的内容 |  |
| icon | String | 否 | 图标，有效值 "success", "loading" |  |
| image | String | 否 | 自定义图标的本地路径，image 的优先级高于 icon | [1.1.0](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html) |
| duration | Number | 否 | 提示的延迟时间，单位毫秒，默认：1500 |  |
| mask | Boolean | 否 | 是否显示透明蒙层，防止触摸穿透，默认：false |  |
| success | Function | 否 | 接口调用成功的回调函数 |  |
| fail | Function | 否 | 接口调用失败的回调函数 |  |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |  |

**示例代码：**

wx.showToast({

 title: '成功',

 icon: 'success',

 duration: 2000

})

### wx.showLoading(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

显示 loading 提示框, 需主动调用 [wx.hideLoading](https://mp.weixin.qq.com/debug/wxadoc/dev/api/wxhideloading) 才能关闭提示框

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| title | String | 是 | 提示的内容 |
| mask | Boolean | 否 | 是否显示透明蒙层，防止触摸穿透，默认：false |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

### wx.hideToast()

隐藏消息提示框

### wx.hideLoading()

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

隐藏 loading 提示框

wx.showLoading({

 title: '加载中',

})

setTimeout(function(){

 wx.hideLoading()

},2000)

### wx.showModal(OBJECT)

​显示模态弹窗

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| title | String | 是 | 提示的标题 |
| content | String | 是 | 提示的内容 |
| showCancel | Boolean | 否 | 是否显示取消按钮，默认为 true |
| cancelText | String | 否 | 取消按钮的文字，默认为"取消"，最多 4 个字符 |
| cancelColor | HexColor | 否 | 取消按钮的文字颜色，默认为"#000000" |
| confirmText | String | 否 | 确定按钮的文字，默认为"确定"，最多 4 个字符 |
| confirmColor | HexColor | 否 | 确定按钮的文字颜色，默认为"#3CC51F" |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数说明：**

| 参数 | 类型 | 说明 | 最低版本 |
| --- | --- | --- | --- |
| confirm | Boolean | 为 true 时，表示用户点击了确定按钮 |  |
| cancel | Boolean | 为 true 时，表示用户点击了取消（用于 Android 系统区分点击蒙层关闭还是点击取消按钮关闭） | [1.1.0](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html) |

**示例代码：**

wx.showModal({

 title: '提示',

 content: '这是一个模态弹窗',

 success: function(res) {

 if (res.confirm) {

 console.log('用户点击确定')

 } else if (res.cancel) {

 console.log('用户点击取消')

 }

 }

})

### wx.showActionSheet(OBJECT)

​显示操作菜单

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| itemList | String Array | 是 | 按钮的文字数组，数组长度最大为6个 |
| itemColor | HexColor | 否 | 按钮的文字颜色，默认为"#000000" |
| success | Function | 否 | 接口调用成功的回调函数，详见返回参数说明 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数说明：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| tapIndex | Number | 用户点击的按钮，从上到下的顺序，从0开始 |

**示例代码：**

wx.showActionSheet({

 itemList: ['A', 'B', 'C'],

 success: function(res) {

 console.log(res.tapIndex)

 },

 fail: function(res) {

 console.log(res.errMsg)

 }

})

#### Bug & Tip

1. bug: Android 6.3.30，wx.showModal 的返回的 confirm 一直为 true；
2. tip: wx.showActionSheet 点击取消或蒙层时，回调 fail, errMsg 为 "showActionSheet:fail cancel"；
3. tip: wx.showLoading 和 wx.showToast 同时只能显示一个，使用 wx.hideToast/wx.hideLoading 都可以关闭提示框；
4. tip: iOS wx.showModal 点击蒙层不会关闭模态弹窗，所以尽量避免使用“取消”分支中实现业务逻辑。

wx.setNavigationBarTitle(OBJECT)

动态设置当前页面的标题。

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| title | String | 是 | 页面标题 |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码：**

wx.setNavigationBarTitle({

 title: '当前页面'

})

wx.showNavigationBarLoading()

在当前页面显示导航条加载动画。

wx.hideNavigationBarLoading()

隐藏导航条加载动画。

### wx.navigateTo(OBJECT)

保留当前页面，跳转到应用内的某个页面，使用wx.navigateBack可以返回到原页面。

**OBJECT 参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| url | String | 是 | 需要跳转的应用内非 tabBar 的页面的路径 , 路径后可以带参数。参数与路径之间使用?分隔，参数键与参数值用=相连，不同参数用&分隔；如 'path?key=value&key2=value2' |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码：**

wx.navigateTo({

 url: 'test?id=1'

})

//test.js

Page({

 onLoad: function(option){

 console.log(option.query)

 }

})

**注意：为了不让用户在使用小程序时造成困扰，我们规定页面路径只能是五层，请尽量避免多层级的交互方式。**

### wx.redirectTo(OBJECT)

关闭当前页面，跳转到应用内的某个页面。

**OBJECT 参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| url | String | 是 | 需要跳转的应用内非 tabBar 的页面的路径，路径后可以带参数。参数与路径之间使用?分隔，参数键与参数值用=相连，不同参数用&分隔；如 'path?key=value&key2=value2' |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码：**

wx.redirectTo({

 url: 'test?id=1'

})

### wx.reLaunch(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

关闭所有页面，打开到应用内的某个页面。

**OBJECT 参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| url | String | 是 | 需要跳转的应用内非 tabBar 的页面的路径 , 路径后可以带参数。参数与路径之间使用?分隔，参数键与参数值用=相连，不同参数用&分隔；如 'path?key=value&key2=value2' |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码：**

wx.reLaunch({

 url: 'test?id=1'

})

//test.js

Page({

 onLoad: function(option){

 console.log(option.query)

 }

})

### wx.switchTab(OBJECT)

跳转到 tabBar 页面，并关闭其他所有非 tabBar 页面

**OBJECT 参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| url | String | 是 | 需要跳转的 tabBar 页面的路径（需在 app.json 的 [tabBar](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/config.html#tabbar) 字段定义的页面），路径后不能带参数 |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码：**

{

 "tabBar": {

 "list": [{

 "pagePath": "index",

 "text": "首页"

 },{

 "pagePath": "other",

 "text": "其他"

 }]

 }

}

wx.switchTab({

 url: '/index'

})

### wx.navigateBack(OBJECT)

关闭当前页面，返回上一页面或多级页面。可通过 [getCurrentPages()](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/app-service/page.html#getCurrentPages()) 获取当前的页面栈，决定需要返回几层。

**OBJECT 参数说明：**

| 参数 | 类型 | 默认值 | 说明 |
| --- | --- | --- | --- |
| delta | Number | 1 | 返回的页面数，如果 delta 大于现有页面数，则返回到首页。 |

**示例代码：**

// 注意：调用 navigateTo 跳转时，调用该方法的页面会被加入堆栈，而 redirectTo 方法则不会。见下方示例代码

// 此处是A页面

wx.navigateTo({

 url: 'B?id=1'

})

// 此处是B页面

wx.navigateTo({

 url: 'C?id=1'

})

// 在C页面内 navigateBack，将返回A页面

wx.navigateBack({

 delta: 2

})

#### Tip

1. tip: wx.navigateTo 和 wx.redirectTo 不允许跳转到 tabbar 页面，只能用 wx.switchTab 跳转到 tabbar 页面

### wx.createAnimation(OBJECT)

创建一个动画实例[animation](https://mp.weixin.qq.com/debug/wxadoc/dev/api/api-animation.html#animation)。调用实例的方法来描述动画。最后通过动画实例的export方法导出动画数据传递给组件的animation属性。

**注意: export 方法每次调用后会清掉之前的动画操作**

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 默认值 | 说明 |
| --- | --- | --- | --- | --- |
| duration | Integer | 否 | 400 | 动画持续时间，单位ms |
| timingFunction | String | 否 | "linear" | 定义动画的效果 |
| delay | Integer | 否 | 0 | 动画延迟时间，单位 ms |
| transformOrigin | String | 否 | "50% 50% 0" | 设置transform-origin |

**timingFunction 有效值：**

| 值 | 说明 |
| --- | --- |
| linear | 动画从头到尾的速度是相同的 |
| ease | 动画以低速开始，然后加快，在结束前变慢 |
| ease-in | 动画以低速开始 |
| ease-in-out | 动画以低速开始和结束 |
| ease-out | 动画以低速结束 |
| step-start | 动画第一帧就跳至结束状态直到结束 |
| step-end | 动画一直保持开始状态，最后一帧跳到结束状态 |

var animation = wx.createAnimation({

 transformOrigin: "50% 50%",

 duration: 1000,

 timingFunction: "ease",

 delay: 0

})

### animation

动画实例可以调用以下方法来描述动画，调用结束后会返回自身，支持链式调用的写法。

**animation 对象的方法列表：**

样式：

| 方法 | 参数 | 说明 |
| --- | --- | --- |
| opacity | value | 透明度，参数范围 0~1 |
| backgroundColor | color | 颜色值 |
| width | length | 长度值，如果传入 Number 则默认使用 px，可传入其他自定义单位的长度值 |
| height | length | 长度值，如果传入 Number 则默认使用 px，可传入其他自定义单位的长度值 |
| top | length | 长度值，如果传入 Number 则默认使用 px，可传入其他自定义单位的长度值 |
| left | length | 长度值，如果传入 Number 则默认使用 px，可传入其他自定义单位的长度值 |
| bottom | length | 长度值，如果传入 Number 则默认使用 px，可传入其他自定义单位的长度值 |
| right | length | 长度值，如果传入 Number 则默认使用 px，可传入其他自定义单位的长度值 |

旋转：

| 方法 | 参数 | 说明 |
| --- | --- | --- |
| rotate | deg | deg的范围-180~180，从原点顺时针旋转一个deg角度 |
| rotateX | deg | deg的范围-180~180，在X轴旋转一个deg角度 |
| rotateY | deg | deg的范围-180~180，在Y轴旋转一个deg角度 |
| rotateZ | deg | deg的范围-180~180，在Z轴旋转一个deg角度 |
| rotate3d | (x,y,z,deg) | 同[transform-function rotate3d](https://developer.mozilla.org/en-US/docs/Web/CSS/transform-function/rotate3d) |

缩放：

| 方法 | 参数 | 说明 |
| --- | --- | --- |
| scale | sx,[sy] | 一个参数时，表示在X轴、Y轴同时缩放sx倍数；两个参数时表示在X轴缩放sx倍数，在Y轴缩放sy倍数 |
| scaleX | sx | 在X轴缩放sx倍数 |
| scaleY | sy | 在Y轴缩放sy倍数 |
| scaleZ | sz | 在Z轴缩放sy倍数 |
| scale3d | (sx,sy,sz) | 在X轴缩放sx倍数，在Y轴缩放sy倍数，在Z轴缩放sz倍数 |

偏移：

| 方法 | 参数 | 说明 |
| --- | --- | --- |
| translate | tx,[ty] | 一个参数时，表示在X轴偏移tx，单位px；两个参数时，表示在X轴偏移tx，在Y轴偏移ty，单位px。 |
| translateX | tx | 在X轴偏移tx，单位px |
| translateY | ty | 在Y轴偏移tx，单位px |
| translateZ | tz | 在Z轴偏移tx，单位px |
| translate3d | (tx,ty,tz) | 在X轴偏移tx，在Y轴偏移ty，在Z轴偏移tz，单位px |

倾斜：

| 方法 | 参数 | 说明 |
| --- | --- | --- |
| skew | ax,[ay] | 参数范围-180~180；一个参数时，Y轴坐标不变，X轴坐标延顺时针倾斜ax度；两个参数时，分别在X轴倾斜ax度，在Y轴倾斜ay度 |
| skewX | ax | 参数范围-180~180；Y轴坐标不变，X轴坐标延顺时针倾斜ax度 |
| skewY | ay | 参数范围-180~180；X轴坐标不变，Y轴坐标延顺时针倾斜ay度 |

矩阵变形：

| 方法 | 参数 | 说明 |
| --- | --- | --- |
| matrix | (a,b,c,d,tx,ty) | 同[transform-function matrix](https://developer.mozilla.org/en-US/docs/Web/CSS/transform-function/matrix) |
| matrix3d |  | 同[transform-function matrix3d](https://developer.mozilla.org/en-US/docs/Web/CSS/transform-function/matrix3d) |

### 动画队列

调用动画操作方法后要调用 step() 来表示一组动画完成，可以在一组动画中调用任意多个动画方法，一组动画中的所有动画会同时开始，一组动画完成后才会进行下一组动画。step 可以传入一个跟 wx.createAnimation() 一样的配置参数用于指定当前组动画的配置。

**示例：**

<view animation="{{animationData}}" style="background:red;height:100rpx;width:100rpx"></view>

Page({

 data: {

 animationData: {}

 },

 onShow: function(){

 var animation = wx.createAnimation({

 duration: 1000,

 timingFunction: 'ease',

 })

 this.animation = animation

 animation.scale(2,2).rotate(45).step()

 this.setData({

 animationData:animation.export()

 })

 setTimeout(function() {

 animation.translate(30).step()

 this.setData({

 animationData:animation.export()

 })

 }.bind(this), 1000)

 },

 rotateAndScale: function () {

 // 旋转同时放大

 this.animation.rotate(45).scale(2, 2).step()

 this.setData({

 animationData: this.animation.export()

 })

 },

 rotateThenScale: function () {

 // 先旋转后放大

 this.animation.rotate(45).step()

 this.animation.scale(2, 2).step()

 this.setData({

 animationData: this.animation.export()

 })

 },

 rotateAndScaleThenTranslate: function () {

 // 先旋转同时放大，然后平移

 this.animation.rotate(45).scale(2, 2).step()

 this.animation.translate(100, 100).step({ duration: 1000 })

 this.setData({

 animationData: this.animation.export()

 })

 }

})

#### bug & tip

1. bug: iOS/Android 6.3.30 通过 step() 分隔动画，只有第一步动画能生效

# 在Canvas上画图

所有在 <canvas/> 中的画图必须用 JavaScript 完成：

WXML：（我们在接下来的例子中如无特殊声明都会用这个 WXML 为模板，不再重复）

<canvas canvas-id="myCanvas" style="border: 1px solid;"/>

JS：（我们在接下来的例子中会将 JS 放在 onLoad 中）

const ctx = wx.createCanvasContext('myCanvas')

ctx.setFillStyle('red')

ctx.fillRect(10, 10, 150, 75)

ctx.draw()

### 第一步：创建一个 Canvas 绘图上下文

首先，我们需要创建一个 Canvas 绘图上下文 CanvasContext。

CanvasContext 是小程序内建的一个对象，有一些绘图的方法：

const ctx = wx.createCanvasContext('myCanvas')

### 第二步：使用 Canvas 绘图上下文进行绘图描述

接着，我们来描述要在 Canvas 中绘制什么内容。

设置绘图上下文的填充色为红色：

ctx.setFillStyle('red')

用 fillRect(x, y, width, height) 方法画一个矩形，填充为刚刚设置的红色：

ctx.fillRect(10, 10, 150, 75)

### 第三步：画图

告诉 <canvas/> 组件你要将刚刚的描述绘制上去：

ctx.draw()

### 结果：



# onPullDownRefresh

在 Page 中定义 onPullDownRefresh 处理函数，监听该页面用户下拉刷新事件。

* 需要在 config 的[window](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/config.html#window)选项中开启 enablePullDownRefresh。
* 当处理完数据刷新后，[wx.stopPullDownRefresh](https://mp.weixin.qq.com/debug/wxadoc/dev/api/pulldown.html#wxstoppulldownrefresh)可以停止当前页面的下拉刷新。

### wx.stopPullDownRefresh()

停止当前页面下拉刷新。

**示例代码：**

Page({

 onPullDownRefresh: function(){

 wx.stopPullDownRefresh()

 }

})

### wx.getExtConfig(OBJECT)

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

获取第三方平台自定义的数据字段。

**OBJECT参数说明：**

| 参数 | 类型 | 必填 | 返回 |
| --- | --- | --- | --- |
| success | Function | 否 | 返回第三方平台自定义的数据 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数说明：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| errMsg | String | 调用结果 |
| extConfig | Object | 第三方平台自定义的数据 |

#### Bug & Tip

1. wx.getExtConfig 暂时无法通过 wx.canIUse 判断是否兼容，开发者需要自行判断 wx.getExtConfig 是否存在来兼容

**示例代码：**

if(wx.getExtConfig) {

 wx.getExtConfig({

 success: function (res) {

 console.log(res.extConfig)

 }

 })

}

### wx.getExtConfigSync()

基础库 1.1.0 开始支持，低版本需做[兼容处理](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html)

获取第三方平台自定义的数据字段的同步接口。

**返回说明：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| extConfig | Object | 第三方平台自定义的数据 |

#### Bug & Tip

1. wx.getExtConfigSync 暂时无法通过 wx.canIUse 判断是否兼容，开发者需要自行判断 wx.getExtConfigSync 是否存在来兼容

**示例代码：**

let extConfig = wx.getExtConfigSync? wx.getExtConfigSync(): {}

console.log(extConfig)

### wx.login(OBJECT)

调用接口获取**登录凭证（code）**进而换取用户登录态信息，包括用户的**唯一标识（openid）** 及本次登录的 **会话密钥（session\_key）**。**用户数据的加解密通讯**需要依赖会话密钥完成。

**OBJECT参数说明：**

| 参数名 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**success返回参数说明：**

| 参数名 | 类型 | 说明 |
| --- | --- | --- |
| errMsg | String | 调用结果 |
| code | String | 用户允许登录后，回调内容会带上 code（有效期五分钟），开发者需要将 code 发送到开发者服务器后台，使用code 换取 session\_key api，将 code 换成 openid 和 session\_key |

**示例代码：**

//app.js

App({

 onLaunch: function() {

 wx.login({

 success: function(res) {

 if (res.code) {

 //发起网络请求

 wx.request({

 url: 'https://test.com/onLogin',

 data: {

 code: res.code

 }

 })

 } else {

 console.log('获取用户登录态失败！' + res.errMsg)

 }

 }

 });

 }

})

### code 换取 session\_key

​ 这是一个 HTTPS 接口，开发者服务器使用**登录凭证 code**获取 session\_key 和 openid。其中 session\_key 是对用户数据进行[加密签名](https://mp.weixin.qq.com/debug/wxadoc/dev/api/signature.html)的密钥。为了自身应用安全，**session\_key 不应该在网络上传输**。

**接口地址：**

https://api.weixin.qq.com/sns/jscode2session?appid=APPID&secret=SECRET&js\_code=JSCODE&grant\_type=authorization\_code

**请求参数：**

| 参数 | 必填 | 说明 |
| --- | --- | --- |
| appid | 是 | 小程序唯一标识 |
| secret | 是 | 小程序的 app secret |
| js\_code | 是 | 登录时获取的 code |
| grant\_type | 是 | 填写为 authorization\_code |

**返回参数：**

| 参数 | 说明 |
| --- | --- |
| openid | 用户唯一标识 |
| session\_key | 会话密钥 |

**返回说明：**

//正常返回的JSON数据包

{

 "openid": "OPENID",

 "session\_key": "SESSIONKEY"

}

//错误时返回JSON数据包(示例为Code无效)

{

 "errcode": 40029,

 "errmsg": "invalid code"

}

### wx.checkSession(OBJECT)

通过上述接口获得的用户登录态拥有一定的时效性。用户越久未使用小程序，用户登录态越有可能失效。反之如果用户一直在使用小程序，则用户登录态一直保持有效。具体时效逻辑由微信维护，对开发者透明。开发者只需要调用wx.checkSession接口**检测当前用户登录态是否有效**。登录态过期后开发者可以再调用wx.login获取新的用户登录态。

**OBJECT参数说明：**

| 参数名 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| success | Function | 否 | 接口调用成功的回调函数，登录态未过期 |
| fail | Function | 否 | 接口调用失败的回调函数，登录态已过期 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

**示例代码：**

wx.checkSession({

 success: function(){

 //session 未过期，并且在本生命周期一直有效

 },

 fail: function(){

 //登录态过期

 wx.login() //重新登录

 ....

 }

})

## 登录态维护

通过 wx.login() 获取到用户登录态之后，需要维护登录态。开发者要注意**不应该直接把 session\_key、openid 等字段作为用户的标识或者 session 的标识**，而应该自己派发一个 session 登录态（请参考登录时序图）。对于开发者自己生成的 session，应该保证其安全性且不应该设置较长的过期时间。session 派发到小程序客户端之后，可将其存储在 storage ，用于后续通信使用。

通过wx.checkSession() 检测用户登录态是否失效。并决定是否调用wx.login() 重新获取登录态

### 登录时序图



#### Bug & Tip

1. bug: iOS/Android 6.3.30，在 App.onLaunch 调用 wx.login 会出现异常；

## 用户数据的签名验证和加解密

### 数据签名校验

为了确保 [开放接口](https://mp.weixin.qq.com/debug/wxadoc/dev/api/open.html) 返回用户数据的安全性，微信会对明文数据进行签名。开发者可以根据业务需要对数据包进行签名校验，确保数据的完整性。

1. 签名校验算法涉及用户的session\_key，通过 [wx.login](https://mp.weixin.qq.com/debug/wxadoc/dev/api/api-login.html#wxloginobject) 登录流程获取用户session\_key，并自行维护与应用自身登录态的对应关系。
2. 通过调用接口（如 [wx.getUserInfo](https://mp.weixin.qq.com/debug/wxadoc/dev/api/open.html)）获取数据时，接口会同时返回 rawData、signature，其中 signature = sha1( rawData + session\_key )
3. 开发者将 signature、rawData 发送到开发者服务器进行校验。服务器利用用户对应的 session\_key 使用相同的算法计算出签名 signature2 ，比对 signature 与 signature2 即可校验数据的完整性。

**如wx.getUserInfo的数据校验：**

接口返回的rawData：

{

 "nickName": "Band",

 "gender": 1,

 "language": "zh\_CN",

 "city": "Guangzhou",

 "province": "Guangdong",

 "country": "CN",

 "avatarUrl": "http://wx.qlogo.cn/mmopen/vi\_32/1vZvI39NWFQ9XM4LtQpFrQJ1xlgZxx3w7bQxKARol6503Iuswjjn6nIGBiaycAjAtpujxyzYsrztuuICqIM5ibXQ/0"

}

用户的 session-key：

HyVFkGl5F5OQWJZZaNzBBg==

所以，用于签名的字符串为：

{"nickName":"Band","gender":1,"language":"zh\_CN","city":"Guangzhou","province":"Guangdong","country":"CN","avatarUrl":"http://wx.qlogo.cn/mmopen/vi\_32/1vZvI39NWFQ9XM4LtQpFrQJ1xlgZxx3w7bQxKARol6503Iuswjjn6nIGBiaycAjAtpujxyzYsrztuuICqIM5ibXQ/0"}HyVFkGl5F5OQWJZZaNzBBg==

使用sha1得到的结果为

75e81ceda165f4ffa64f4068af58c64b8f54b88c

### 加密数据解密算法

接口如果涉及敏感数据（如[wx.getUserInfo](https://mp.weixin.qq.com/debug/wxadoc/dev/api/open.html)当中的 openId 和unionId ），接口的明文内容将不包含这些敏感数据。开发者如需要获取敏感数据，需要对接口返回的**加密数据( encryptedData )**进行对称解密。 解密算法如下：

1. 对称解密使用的算法为 AES-128-CBC，数据采用PKCS#7填充。
2. 对称解密的目标密文为 Base64\_Decode(encryptedData)。
3. 对称解密秘钥 aeskey = Base64\_Decode(session\_key), aeskey 是16字节。
4. 对称解密算法初始向量 为Base64\_Decode(iv)，其中iv由数据接口返回。

微信官方提供了多种编程语言的示例代码（[点击下载](https://mp.weixin.qq.com/debug/wxadoc/dev/demo/aes-sample.zip)）。每种语言类型的接口名字均一致。调用方式可以参照示例。

另外，为了应用能校验数据的有效性，我们会在敏感数据加上数据水印( watermark )

**watermark参数说明：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| watermark | OBJECT | 数据水印 |
| appid | String | 敏感数据归属appid，开发者可校验此参数与自身appid是否一致 |
| timestamp | DateInt | 敏感数据获取的时间戳, 开发者可以用于数据时效性校验 |

如接口[wx.getUserInfo](https://mp.weixin.qq.com/debug/wxadoc/dev/api/open.html)敏感数据当中的watermark：

{

 "openId": "OPENID",

 "nickName": "NICKNAME",

 "gender": GENDER,

 "city": "CITY",

 "province": "PROVINCE",

 "country": "COUNTRY",

 "avatarUrl": "AVATARURL",

 "unionId": "UNIONID",

 "watermark":

 {

 "appid":"APPID",

 "timestamp":TIMESTAMP

 }

}

注：此前提供的**加密数据（encryptData）**以及对应的加密算法将被弃用，请开发者不要再依赖旧逻辑。

### wx.getUserInfo(OBJECT)

获取用户信息，需要先调用 [wx.login](https://mp.weixin.qq.com/debug/wxadoc/dev/api/api-login.html#wxloginobject) 接口。

**OBJECT参数说明：**

| 参数名 | 类型 | 必填 | 说明 | 最低版本 |
| --- | --- | --- | --- | --- |
| withCredentials | Boolean | 否 | 是否带上登录态信息 | [1.1.0](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html) |
| success | Function | 否 | 接口调用成功的回调函数 |  |
| fail | Function | 否 | 接口调用失败的回调函数 |  |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |  |

**注：当 withCredentials 为 true 时，要求此前有调用过 wx.login 且登录态尚未过期，此时返回的数据会包含 encryptedData, iv 等敏感信息；当 withCredentials 为 false 时，不要求有登录态，返回的数据不包含 encryptedData, iv 等敏感信息。**

**success返回参数说明：**

| 参数 | 类型 | 说明 |
| --- | --- | --- |
| userInfo | OBJECT | 用户信息对象，不包含 openid 等敏感信息 |
| rawData | String | 不包括敏感信息的原始数据字符串，用于计算签名。 |
| signature | String | 使用 sha1( rawData + sessionkey ) 得到字符串，用于校验用户信息，参考文档 [signature](https://mp.weixin.qq.com/debug/wxadoc/dev/api/signature.html)。 |
| encryptedData | String | 包括敏感数据在内的完整用户信息的加密数据，详细见[加密数据解密算法](https://mp.weixin.qq.com/debug/wxadoc/dev/api/signature.html#加密数据解密算法) |
| iv | String | 加密算法的初始向量，详细见[加密数据解密算法](https://mp.weixin.qq.com/debug/wxadoc/dev/api/signature.html#加密数据解密算法) |

**示例代码：**

wx.getUserInfo({

 success: function(res) {

 var userInfo = res.userInfo

 var nickName = userInfo.nickName

 var avatarUrl = userInfo.avatarUrl

 var gender = userInfo.gender //性别 0：未知、1：男、2：女

 var province = userInfo.province

 var city = userInfo.city

 var country = userInfo.country

 }

})

encryptedData 解密后为以下 json 结构，详见[加密数据解密算法](https://mp.weixin.qq.com/debug/wxadoc/dev/api/signature.html#加密数据解密算法)

{

 "openId": "OPENID",

 "nickName": "NICKNAME",

 "gender": GENDER,

 "city": "CITY",

 "province": "PROVINCE",

 "country": "COUNTRY",

 "avatarUrl": "AVATARURL",

 "unionId": "UNIONID",

 "watermark":

 {

 "appid":"APPID",

 "timestamp":TIMESTAMP

 }

}

#### UnionID机制说明：

如果开发者拥有多个移动应用、网站应用、和公众帐号（包括小程序），可通过unionid来区分用户的唯一性，因为只要是同一个微信开放平台帐号下的移动应用、网站应用和公众帐号（包括小程序），用户的unionid是唯一的。换句话说，同一用户，对同一个微信开放平台下的不同应用，unionid是相同的。

**微信开放平台绑定小程序流程**

前提：微信开放平台帐号必须已完成开发者资质认证

开发者资质认证流程：

登录微信开放平台(open.weixin.qq.com) – 帐号中心 – 开发者资质认证



绑定流程：

登录微信开放平台（open.weixin.qq.com）—管理中心—公众帐号—绑定公众帐号



#### Bug & Tip

1. tip: wx.getUserInfo 接口需要用户授权，请兼容用户拒绝授权的场景。

### wx.requestPayment(OBJECT)

发起微信支付。

**Object参数说明：**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| timeStamp | String | 是 | 时间戳从1970年1月1日00:00:00至今的秒数,即当前的时间 |
| nonceStr | String | 是 | 随机字符串，长度为32个字符以下。 |
| package | String | 是 | 统一下单接口返回的 prepay\_id 参数值，提交格式如：prepay\_id=\* |
| signType | String | 是 | 签名算法，暂支持 MD5 |
| paySign | String | 是 | 签名,具体签名方案参见[微信公众号支付帮助文档](https://pay.weixin.qq.com/wiki/doc/api/jsapi.php?chapter=4_3); |
| success | Function | 否 | 接口调用成功的回调函数 |
| fail | Function | 否 | 接口调用失败的回调函数 |
| complete | Function | 否 | 接口调用结束的回调函数（调用成功、失败都会执行） |

了解更多信息，请查看[微信支付接口文档](https://pay.weixin.qq.com/wiki/doc/api/wxa/wxa_api.php?chapter=7_3&index=1)

**回调结果：**

| 回调类型 | errMsg | 说明 |
| --- | --- | --- |
| success | requestPayment:ok | 调用支付成功 |
| fail | requestPayment:fail cancel | 用户取消支付 |
| fail | requestPayment:fail (detail message) | 调用支付失败，其中 detail message 为后台返回的详细失败原因 |

**示例代码：**

wx.requestPayment({

 'timeStamp': '',

 'nonceStr': '',

 'package': '',

 'signType': 'MD5',

 'paySign': '',

 'success':function(res){

 },

 'fail':function(res){

 }

})

#### Bug & Tip

1. bug: 6.5.2 及之前版本中，用户取消支付不会触发 fail 回调，只会触发 complete 回调，回调 errMsg 为 'requestPayment:cancel'