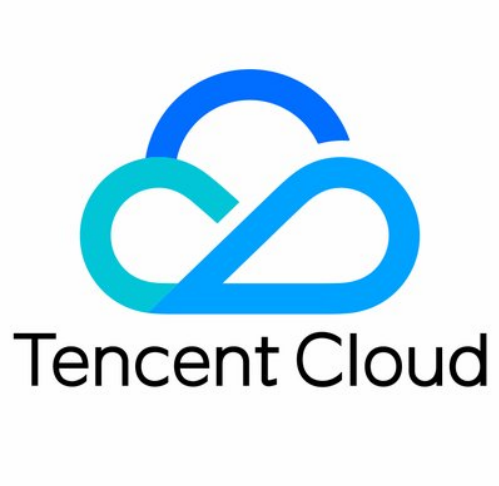
Tencent Cloud Super App as a Service

Guildlines for Mini Program Development

Product Documentation



©2013-2025 Tencent Cloud International Pte. Ltd.

Copyright Notice

©2013-2025 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice

All trademarks associated with Tencent Cloud and its services are owned by the Tencent corporate group, including its parent, subsidiaries and affiliated companies, as the case may be. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

Guildlines for Mini Program Development
- Mini Program Introduction and Development Environment
- Mini Program Code Composition
  - JSON Configuration
  - WXML Template
  - WXSS Style
  - JavaScript
- Guidance
  - Learn About Mini Programs
    - Mini Program Host Environment
    - Operation Mechanism
    - Code Structure
    - Directory Structure
  - Configuring Mini Program
    - Global and Page Configuration
  - Mini Program Framework
    - Introduce
    - Logic Layer
    - View Layer
  - Custom Components
  - Basic Competencies
    - Network
    - Storage
    - File System
    - Subpackage Loading
    - Canvas
    - Custom TabBar
    - Multi-Threaded Worker
    - DarkMode Adaptation Guide
  - Open Capabilities
    - Banner Ad
    - Rewarded Video Ad
- Framework
  - Mini Program Configuration
    - Project Configuration

©2013-2025 Tencent Cloud International Pte. Ltd.

Global Configuration
Page Configuration
Framework Interface
App
Page
Custom Components
Modularity
Basic Function
WXML Syntax Reference
Introduce
Data Binding
List Rendering
Conditional Rendering
Templates
Referencing
WXS Syntax Reference
Introduce
WXS Modules
Variables
Annotations
Operators
Statements
Data Types
Base Library
Availability
Components
Component Overview
Native Components
View Container
Fundamentals
Map Component
Media Component
Open Capabilities
Canvas
Navigation Component
Form Component
API
API Overview

- Basic
- Routing
- Navigation
- Share
- Interface
  - Window
  - Font
  - Interaction
  - Navigation Bar
  - Background
  - TabBar
  - Pull-down refresh
  - Scroll
  - Animation
  - Custom component
  - Menu
- Network
  - UDP Communication
  - TCP Communication
  - mDNS
  - Initiate request
  - Upload
  - Websocket
  - Download
- Payment
- Data Caching
- Data Analysis
- Canvas
  - Canvas Overview
  - OffscreenCanvas
  - Path2D
  - Canvas
  - canvasContext
  - Color
- Worker
- Media
  - Rich Text
  - Video

©2013-2025 Tencent Cloud International Pte. Ltd.

- Audio Recording
- Background Audio
- Map
- Real-Time Audio/Video
- Transparent Video
- Camera
- Image
- Audio

WXML

File

Device

- Contact
- Vibration
- Battery
- Screen
- Memory
- Device Motion
- Accessibility
- Network
- Keyboard
- Phone Call
- Accelerometer
- Compass
- Gyroscope
- SMS
- Code Scanning
- Encryption
- NFC—Ndef
- NFC—NfcA
- NFC—NfcB
- NFC—NfcF
- NFC—NfcV
- WiFi
- Calendar
- Clipboard
- NFC—getNFCAdapter
- NFC—NFCAdapter
- NFC—IsoDep

©2013-2025 Tencent Cloud International Pte. Ltd.

NFC—MifareClassic
NFC—MifareUltralight
Bluetooth - Low Energy Central Device
Bluetooth - Low Energy Peripheral Device
Bluetooth - Beacon
Bluetooth - General
Location
Open APIs
Custom APIs
Sensitive APIs
H5 Real-Time Communication
Ad
Server Backend
Mini Program Login
Subscription Message
Mini Program Payment
Error Code
JS SDK
web-view
API Description
Base Library
Base Library Introduction
Base Library Maintenance
IDE Operation Instructions
Overview
Interfaces
Settings
Code Editing
Mini Program Debugging
Additional Information

©2013-2025 Tencent Cloud International Pte. Ltd.

Guildlines for Mini Program Development
Mini Program Introduction and Development Environment

Last updated：2025-03-20 18:01:55

Mini programs are a new way to connect users and services. They can be easily accessed and distributed within an application, and provide an outstanding user experience. Developers can easily develop and publish a mini program after learning.

1. Hello World

Before introducing the history and technical details of mini programs, follow our steps to develop this Hello World.
Step 1: Go to the corresponding to your own operating system to download and install the installation package.
Step 2: Open the developer tool and click New Mini program Project. Leave alone the AppID and select No AppID when you create a new project and uncheck the option of "Create general quick startup template".
Step 3: Add necessary code.
Create `app.json` in the root directory, which reads as follows:

```
  "pages": ["pages/index/index"]

}
```

Create a new pages directory under the root directory, then create a new index directory under the pages directory, and then create two files index.wxml and index.js under the index directory.
The content of index.wxml: <text>Hello World</text>
The content of index.js: Page({})
By writing these lines of code, Hello World appears in the simulator of the developer tool.

2. Mini program introduction

2.1 History of mini program

From a technical perspective, mini programs are not a concept that comes out of nowhere. When WebView became an important entry point for the mobile Web, the JS API came into being.

```
WeixinJSBridge.invoke('imagePreview', {
```

```
    current: 'http://inews.gtimg.com/newsapp_bt/0/1693121381/641',
    urls: [ // URL list of all images, array format
        'https://img1.gtimg.com/10/1048/104857/10485731_980x1200_0.jpg',
        'https://img1.gtimg.com/10/1048/104857/10485726_980x1200_0.jpg',
        'https://img1.gtimg.com/10/1048/104857/10485729_980x1200_0.jpg'
    ]
}, function(res) {
    console.log(res.err_msg)
})
```

This JS API calls the native components to browse the image. Compared to the introduction of an additional JS image preview component library, this method is very simple and efficient.

In fact, WeChat officials have not exposed such calls to users outside the Tencent company. This kind of API was initially provided to their internal business usage. After many external developers discovered that, they tried to use it, gradually becoming the actual standard of WeChat's web pages.

The platform has a complete set of built-in web development toolkit, named JS-SDK, including dozens of APIs such as shooting, recording, voice recognition, QR code, map, payment, sharing, vouchers, etc. It opens up a whole new world for Web developers to use native capabilities within the host application to do things that were previously impossible or difficult to do.

Call native images in the following way:

```
wx.previewImage({
   current: 'https://img1.gtimg.com/10/1048/104857/10485726_980x1200_0.jpg',
   urls: [ // URL list of all images, array format
     'https://img1.gtimg.com/10/1048/104857/10485731_980x1200_0.jpg',
     'https://img1.gtimg.com/10/1048/104857/10485726_980x1200_0.jpg',
     'https://img1.gtimg.com/10/1048/104857/10485729_980x1200_0.jpg'
   ],
   success: function(res) {
     console.log(res)
   }
})
```

JS-SDK solves the problem that mobile web page's capabilities are insufficient by exposing the host application APIs, allowing Web developers to have more capabilities. However, it does not fix the poor experience issues for mobile web pages.

When a user accessed a web page, a blank white screen would occur before the browser started displaying. And this problem became worse on the mobile platforms because of device performance and network speed. We focused on solving this problem by providing an enhanced version of JS-SDK, which includes an important feature named offline storage for Web resources.

This feature is similar to Application Cache of HTML5, but avoids some shortcomings.

In the internal test, we found that offline storage can solve some problems, but for some complex pages, the white screen issue still existed. For example, for pages loaded with a large number of CSS or JavaScript files, the execution

time of these files take up a large number of UI threads. Under this circumstances, even though the offline storage can speed up resource loading, the white screen still displays. Additionally, the file-based Cache is more complicated when the code file is updated, and the capability requirement for developers is higher.

Except for the white screen, the poor web experience problems also include delayed page response, mainly manifested in two aspects: unsmooth page switch and delayed click responses.

For some experienced Web developers, they will use SPA framework to simulate native page switching in the client. They usually simulate multiple pages in a WebView, through CSS processing, combined with the refined script code to achieve smooth click response and page switching. However, not all developers have the time and energy to make the page experience great.

The problem that the host application faces is how to design a better system, so that all developers can get a better experience in the host application. This problem cannot be solved by previous JS-SDK and a new system is required for web developers to achieve:

Fast loading

Powerful capabilities

Native experience

Easy-to-use and secure data openness

Efficient and simple development

This system is the small program that will be explained in detail in this book.

2.2. Difference between mini program development and ordinary web page development

The primary programing language for mini programs is JavaScript, so mini program development is often compared to the traditional web page development. The two share many similarities, making it relatively easy for front-end developers to transit from the web development to mini program development. However, there are still some differences:

In web development, the rendering and scripting threads are mutually exclusive, which is why long-running scripts can cause unresponsive pages. In mini programs, however, the two are separate and run in separate threads. Web developers can use various DOM APIs exposed by browsers for DOM selection and manipulation. As mentioned above, the logical layer and rendering layer of the mini programs are separate, with the logical layer running in JSCore without a complete browser object, resulting in a lack of relevant DOM and BOM APIs. This difference means that some familiar libraries for front-end developers, such as jQuery and Zepto, cannot run in mini programs. At the same time, JSCore environment is different from the NodeJS environment, so some NPM package cannot run in mini programs either.

Web developers need to deal with various browser environment, such as IE, Chrome, QQ browser on the PC side, and Safari, Chrome and various WebViews in iOS and Android systems on the mobile side. For mini program developers, they mainly need to deal with the host application clients of the two major operating systems: iOS and Android, and mini program development tools. These three major running environments of mini programs are also different, as shown in the following table:

Operating environment	logic layer	render layer

iOS	JavaScriptCore	WKWebView
Android	X5 JSCore	X5 browser
IDE	NWJS	Chrome WebView

When web developers develop web pages, they only need to use a browser and use some auxiliary tools or editors. The development of mini programs is different. It needs to go through the process of applying for a mini program account, installing mini program developer tools, configuring projects, etc.

3. Mini program features

For ordinary users, mini programs make application accessible easily. Users can directly open them by scanning QR code, searching or a share from friends. The mini programs enhance service providers customer outreach with excellent user experience.
For developers, the fast loading and rendering capabilities of the mini program framework, along with the accompanying cloud capabilities, operational, and data aggregation capabilities, allow developers to focus on specific business logic development without dealing with trivial tasks.

4. Preparing for mini program development

Developers can automatically generate the corresponding App ID after creating a mini program on the platform, and provide different versions of IDE download addresses. Developers can choose the appropriate version to download.

©2013-2025 Tencent Cloud International Pte. Ltd.

Mini Program Code Composition
JSON Configuration

Last updated：2024-11-21 18:28:34

JSON is a data format, not a programming language. For mini programs, JSON plays the role of static configuration.

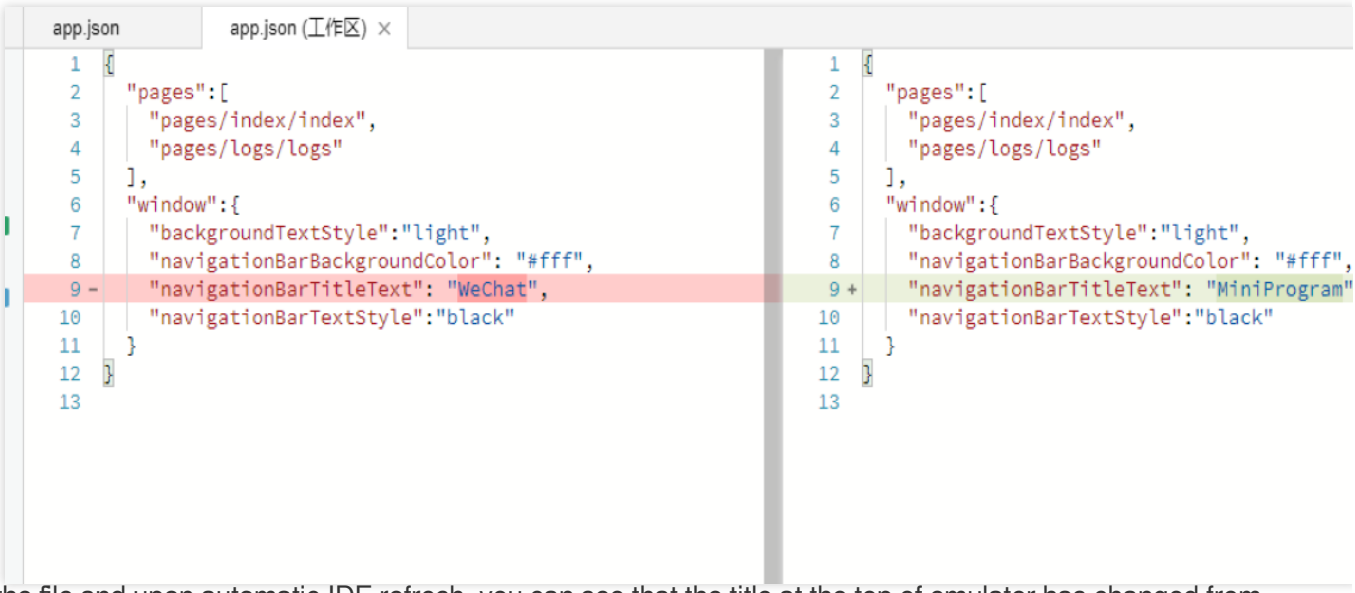
Example

Let's start by looking at an example. Please find the app.json file in the root directory and open it in the code editor. And the app.json file code for this document is as follows:

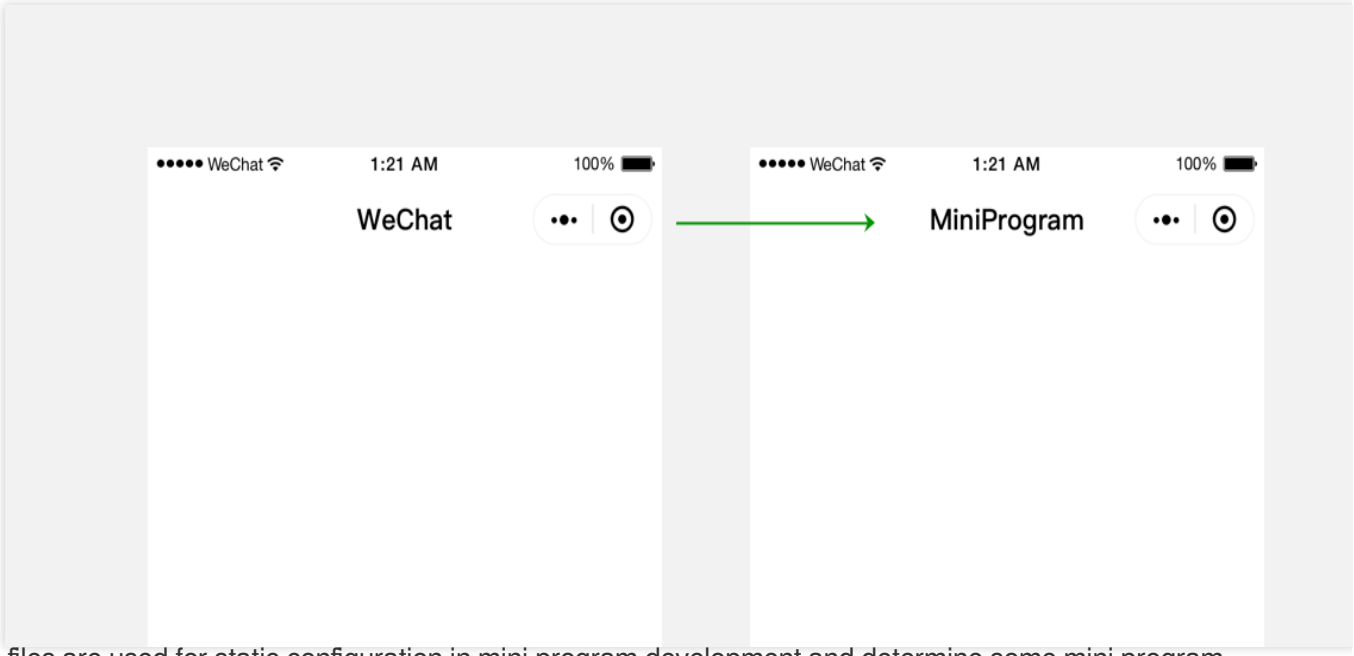
```
{
  "pages":[
    "pages/index/index",
    "pages/logs/logs"
  ],
  "window":{
    "backgroundTextStyle":"light",
    "navigationBarBackgroundColor": "#fff",
    "navigationBarTitleText": "WeChat",
    "navigationBarTextStyle":"black"
  }
}
```

Change the value of "navigationBarTitleText" in line 9 from WeChat to MiniProgram.

©2013-2025 Tencent Cloud International Pte. Ltd.



Save the file and upon automatic IDE refresh, you can see that the title at the top of emulator has changed from WeChat to MiniProgram.



JSON files are used for static configuration in mini program development and determine some mini program performance before it runs. Please note that the mini program cannot dynamically update the JSON configuration file during the running process in order to make corresponding changes.

JSON syntax

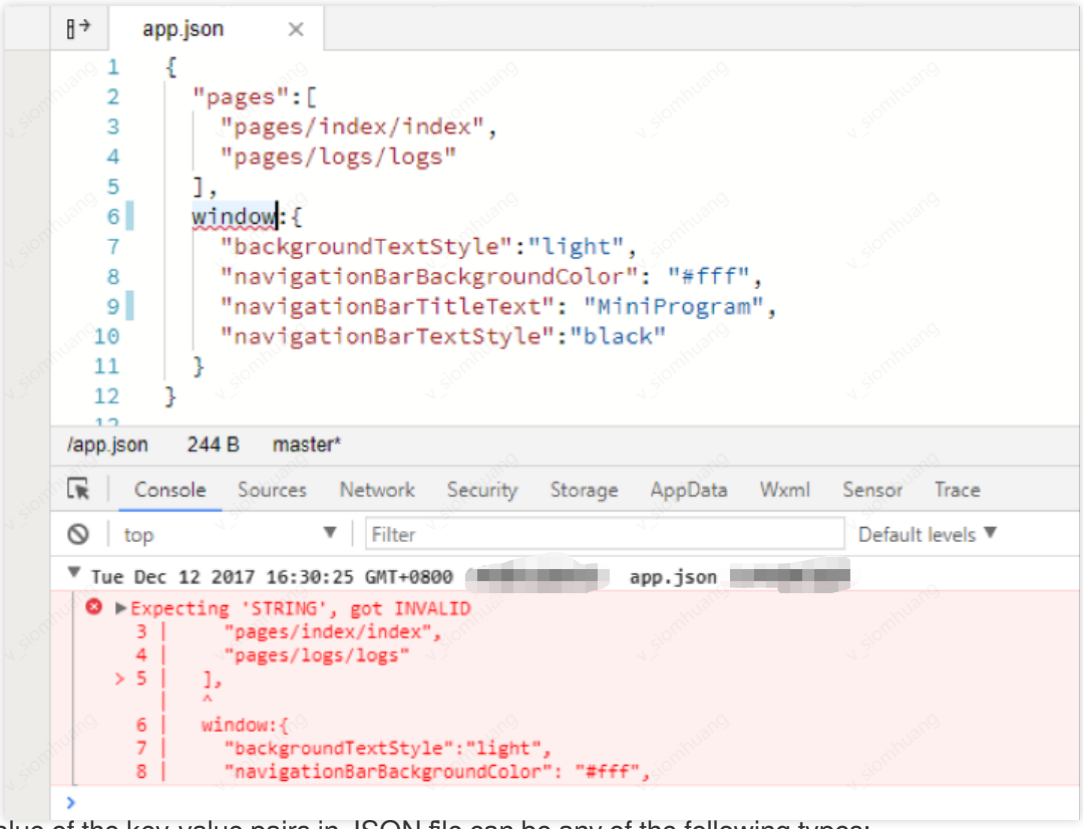
Compared with XML file, JSON file is a lightweight data exchange format, as it usually doesn't require specialized tools for reading or modifying.

JSON file code is wrapped in curly brackets {} and the data is in a key-value format.

```
"pages":[
    "pages/index/index"
    "pages/logs/logs"
],
"window" :{
    "backgroundTextstyle":"light"
    "navigationBarBackgroundColor":"#fff"
    "navigationBarTitleText":"Wechat"
    "navigationBarTextstyle":"black"
    }
}
```

It looks like JavaScript's object expression, but it's different.

The keys in JSON file must be double-quoted strings. Common mistakes include forgetting to enclose the keys in double quotes or mistakenly enclose them in single quotes.

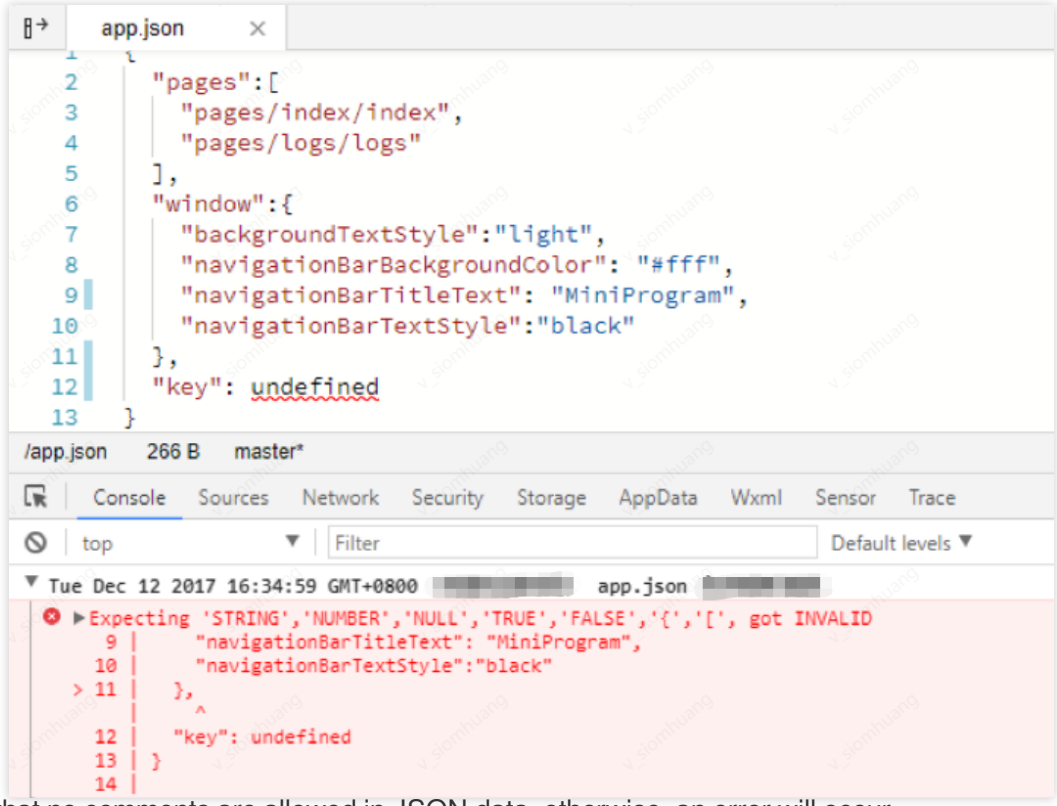


The value of the key-value pairs in JSON file can be any of the following types:

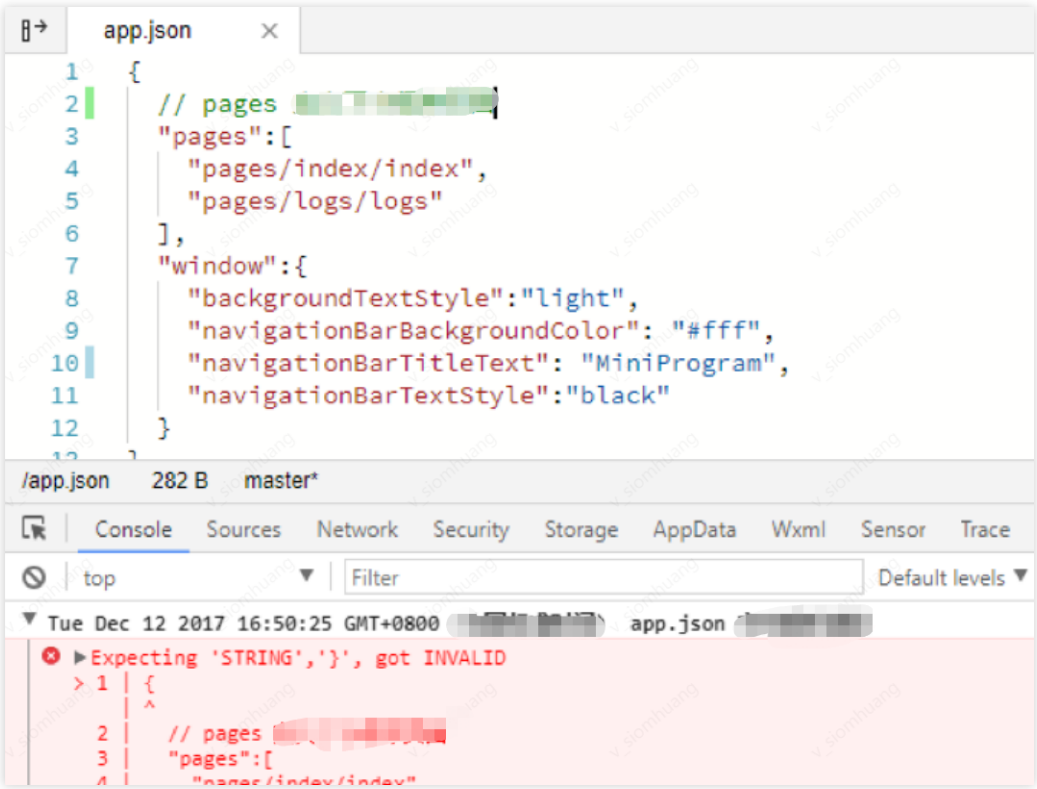
1. Number, including floating-point numbers and integers.
2. String that enclosed in double quotes.
3. Boolean, true or false.
4. Array that enclosed in square brackets [].
5. Object that enclosed in curly brackets {}.
6. Null.

Other value types will trigger an error, for example, in JavaScript.

©2013-2025 Tencent Cloud International Pte. Ltd.



Note that no comments are allowed in JSON data, otherwise, an error will occur.



WXML Template

Last updated：2024-11-21 18:28:34

WeiXin Markup Language (WXML) is a set of markup language for framework design. It can be used with basic components and event systems to construct page structures.
Find the app.json file in the root directory and open it in the IDE. Add a new line `pages/wxml/index` above the line `pages/index/index` and save file. After the simulator is refreshed automatically, you can find the `pages/wxml/index.WXML` file in the editor. This document describes how to modify this file.

Overview

The suffix of the WXML file is `.wxml` . Open the pages/wxml/index.WXML file and you will find that the WXML statements are similar to the HTML syntax if you are familiar with the HTML coding.

```
<!--pages/wxml/index.wxml-->

<text>pages/wxml/index.wxml</text>
```

The basic syntax of WXML without any logical features is as follows:

```
<!-- Add comment here-->

<Tag name Attribute name 1="Attribute value 1" Attribute name 2="Attribute value 2"
```

A complete WXML statement consists of an opening tag and a closing tag, which can wrap either content or other WXML statements. This is similar to HTML. The difference is that WXML statements must be strictly closed, otherwise, a compilation error might occur.

```
<text>hello world


<!--
text is not closed, resulting in a compile error:
VM148:2 ./pages/wxml/index.wxml
 end tag missing, near text
> 1 | <text>hello world
    | ^
-->
```

Tags can have attributes that provide more information about the WXML element. Attributes are always defined in the opening tag. Most of the attributes are in key="value" format except for some special ones. Note that attributes in

WXML are case-sensitive, which means that class and Class are different attributes in WXML. The following is an example of text tags:

```
<!-- A simple text tag-->
<text>hello world</text>

<!-- view contains text tags -->
<view>
  <text>hello world</text>
</view>
```

An example of image tags with attributes is shown below:

```
<image class="userinfo-avatar" src="./image/a.png" ></image>
```

Data binding

UI page displays must be dynamically updated in response to data variations at specific times or alterations caused by user actions. This necessitates incorporating dynamic page rendering capabilities during mini program execution. In Web development, developers use JavaScript to update pages in real time through DOM manipulation API. In mini program, use this data binding provided by the WXML to implement this.
Here is an example:
Modify the content of the pages/wxml/index.wxml file a little bit, as shown below:

```
<!--pages/wxml/index.wxml-->
<text>Current time: {{time}}</text>
```

After the file is saved and refreshed in the IDE, the simulator does not display the current time, because we have not set any initial value for time. Open the pages/wxml/index.js file and add `time:` in the data curly brackets `(new Date()).toString()`, as shown below:

```
// pages/wxml/index.js
Page({
  /**
   * Initial page data
   */
  data: {
    time: (new Date()).toString()
  },
})
```

After saving and refreshing, simulator displays the correct current time, and the compile time will be updated.

WXML binds the data object attributes in the WXML file and the corresponding JavaScript file by {{variable name}}. In order to keep it simple, the following format is used to show this coding logic. You can use the first comment part to show the data structure in the script file corresponding to WXML, as shown below:

```
<!--
{
  time: (new Date()).toString()
}
-->
<text> Current time: {{time}}/text>
```

Attribute values can also be changed dynamically and must be wrapped in double quotes, as shown below:

```
<!-- Correct coding -->
<text data-test="{{test}}"> hello world</text>


<!-- Wrong coding  -->
<text data-test={{test}}> hello world </text >
```

Please note that the variable is case-sensitive, that is, {{name}} and {{Name}} are two different variables, as shown below:

```
<!--
{
  w: 'w',
  W: 'W'
}
-->


<view>{{w}}</view>
<view>{{W}}</view>


<!-- Output
w
W
-->
```

Note that variables or variables is set to will not be synchronized to the wxml, as shown below:

```
<!--
{
  var2: undefined,
  var3: null,
  var4: "var4"
```

```
}
-->


<view>{{var1}}</view>
<view>{{var2}}</view>
<view>{{var3}}</view>
<view>{{var4}}</view>


<!--
Output：
null
var4
-->
```

For more details about data binding, see Chapter 3.

Logical syntax

You can implement dynamic rending through {{variable name}}in WXML and implement simple operations in {{ }}.

Ternary operation:

```
<!-- Display page content conditionally based on whether a is equal to 10 or not --
<text>{{ a === 10? "variable a is equal to 10": "variable a is not equal to 10"}}</
```

Arithmetic operation:

```
<!--
{ a: 1,  b: 2, c: 3 }
-->


<view> {{a + b}} + {{c}} + d </view>


<!-- Output 3 + 3 + d -->
```

Similar to arithmetic operation, it also supports string concatenation, as shown below:

```
<!--
{ name: 'world' }
-->
```

```
<view>{{"hello " + name}}</view>


<!-- Output hello world -->
```

You can also place numbers, strings or arrays in the {{ }}, as shown below:

```
<text>{{[1,2,3]}}</text>

<!-- Output 1,2,3 -->



<text>{{"hello world"}}</text>

<!-- Output hello world -->
```

Logical judgement

In WXML, use `wx:if="{{condition}}"` to determine whether or not to render this code block:

```
<view wx:if="{{condition}}"> True </view>
```

Use wx:elif and wx:else to add an "else" block:

```
<view wx:if="{{length > 5}}"> 1 </view>
<view wx:elif="{{length > 2}}"> 2 </view>
<view wx:else> 3 </view>
```

Because `wx:if` is a control attribute, you must add it to a tag. To judge multiple component tags at once, you can use a `<block/>` tag to package multiple components together and use the `wx:if` control attribute above.

```
<block wx:if="{{true}}">
  <view> view1 </view>
  <view> view2 </view>
</block>
```

List rendering

In a component, by binding an array using the `wx:for` control attribute, you can use the data of the array items to re-render this component. The subscript variable name of the current item of the default array defaults to `index` ,

and the variable name of the current item of the array defaults to `item` :

```
<!-- `array` is an array -->
<view wx:for="{{array}}">
  {{index}}: {{item.message}}
</view>

<!-- Corresponding script file
Page({
  data: {
    array: [{
      message: 'foo',
    }, {
      message: 'bar'
    }]
  }
})
-->
```

You can use `wx:for-item` to specify the variable name of the array's current element and use `wx:for-index` to specify the variable name of the array's current subscript:

```
<view wx:for="{{array}}" wx:for-index="idx" wx:for-item="itemName">
  {{idx}}: {{itemName.message}}
</view>
```

Similar to `block wx:if` , you can use `wx:for` on a `<block/>` tag to render a structural block containing multiple nodes:

```
<block wx:for="{{[1, 2, 3]}}">
  <view> {{index}}: </view>
  <view> {{item}} </view>
</block>
```

If the positions of list items dynamically change or new items are added to the list and you want the items in the list to retain their features and statuses (such as the content input in <input/> and the selection status of <switch/>), you must use `wx:key` to specify the unique identifiers of the items in the list.

The `wx:key` value is provided in either of the following formats:

1. String: Represents an attribute of an item in the for loop array. The value of this attribute must be a unique string or number in the list and cannot dynamically change.

2. Reserved keyword `this` : Represents the item itself in the for loop. This indicates that the item itself is a unique string or number. For example:

When a data change triggers re-rendering at the rendering layer, the components that have keys are corrected. The framework ensures that they are reordered, rather than recreated. This ensures the components retain their statuses

©2013-2025 Tencent Cloud International Pte. Ltd.

and improves the efficiency of list rendering.

```
<switch wx:for="{{objectArray}}" wx:key="unique" > {{item.id}} </switch>
<button bindtap="switch"> Switch </button>
<button bindtap="addToFront"> Add to the front </button>


<switch wx:for="{{numberArray}}" wx:key="*this" > {{item}} </switch>
<button bindtap="addNumberToFront"> Add Number to the front </button>
```

```
Page({
  data: {
    objectArray: [
      {id: 5, unique: 'unique_5'},
      {id: 4, unique: 'unique_4'},
      {id: 3, unique: 'unique_3'},
      {id: 2, unique: 'unique_2'},
      {id: 1, unique: 'unique_1'},
      {id: 0, unique: 'unique_0'},
    ],
    numberArray: [1, 2, 3, 4]
  },
  switch: function(e) {
    const length = this.data.objectArray.length
    for (let i = 0; i < length; ++i) {
      const x = Math.floor(Math.random() * length)
      const y = Math.floor(Math.random() * length)
      const temp = this.data.objectArray[x]
      this.data.objectArray[x] = this.data.objectArray[y]
      this.data.objectArray[y] = temp
    }
    this.setData({
      objectArray: this.data.objectArray
    })
  },
  addToFront: function(e) {
    const length = this.data.objectArray.length
    this.data.objectArray = [{id: length, unique: 'unique_' + length}].concat(this.
    this.setData({
      objectArray: this.data.objectArray
    })
  },
  addNumberToFront: function(e){
    this.data.numberArray = [ this.data.numberArray.length + 1 ].concat(this.data.n
    this.setData({
      numberArray: this.data.numberArray
```

```
    })
  }
})
```

Templates

WXML provides templates, where you can define code snippets and then call them in different places. Use the `name` attribute to specify the name of the template. Then define that code snippet inside <template/>:

```
<template name="msgItem">
  <view>
    <text> {{index}}: {{msg}} </text>
    <text> Time: {{time}} </text>
  </view>
</template>
```

Use the `is` attribute to declare the template to be used. Then, pass the data needed by the template. For example:

```
<!--
item: {
  index: 0,
  msg: 'this is a template',
  time: '2016-06-18'
}
-->


<template name="msgItem">
  <view>
    <text> {{index}}: {{msg}} </text>
    <text> Time: {{time}} </text>
  </view>
</template>


<template is="msgItem" data="{{...item}}"/>

<!-- Output
0: this is a template Time: 2016-06-18
-->
```

Use `is` to dynamically decide which template to be rendered, as shown below:

```
<template name="odd">
```

```
  <view> odd </view>
</template>


<template name="even">
  <view> even </view>
</template>


<block wx:for="{{[1, 2, 3, 4, 5]}}">
  <template is="{{item % 2 == 0 ? 'even' : 'odd'}}"/>
</block>



<!-- Output
odd
even
odd
even
odd
-->
```

Reference

WXML provides two referencing methods `import` and `include`.

`import` can use the template defined by the target file in the current file, for example, a template named `item` is defined in `item.wxml`:

```
<!-- item.wxml -->
<template name="item">
  <text>{{text}}</text>
</template>
```

By referencing `item.wxml` in `index.wxml`, you can use the `item` template:

```
<import src="item.wxml"/>

<template is="item" data="{{text: 'forbar'}}"/>
```

The concept of "scope" is used for `import` references. This means that the reference will import the template defined in the target file but not the template imported by the target file.

For example, C references B and B references A, and the template defined by B can be used in C, and the template defined by A can be used in B, but C cannot use the template defined by A, as shown below:

```
<!-- A.wxml -->
<template name="A">
  <text> A template </text>
</template>
```

```
<!-- B.wxml --><import src="a.wxml"/><template name="B">  <text> B template </text>
```

```
<!-- C.wxml -->
<import src="b.wxml"/>

<template is="A"/>  <!-- An alarm will be triggered, as template A is not defined i

<template is="B"/>
```

`include` can introduce the entire code excluding `<template/> <wxs/>` . This is equivalent to copying the code to the location of `include` . For example:

```
<!-- index.wxml -->
<include src="header.wxml"/>

<view> body </view>

<include src="footer.wxml"/>
```

```
<!-- header.wxml -->
<view> header </view>
```

```
<!-- footer.wxml -->
<view> footer </view>
```

Common attributes

The attributes that are supported by all wxml tags are common attributes.

Attribute	Type	Definition	Description
id	String	Unique identifier of the component	Unique for entire page

class	String	Style class for the component	The style class defined in the corresponding WXSS
style	String	Inline style of the component	The inline style that can be set dynamically
hidden	Boolean	Whether the component is displayed	Default display for all components
data-*	Any	Custom attribute	When an event is triggered on the component, it will be sent to the event handler function.
bind*/catch*	EventHandler	Component Event	-

WXSS Style

Last updated：2025-03-25 18:15:54

WeChat Style Sheets (WXSS) is a style language used to describe the style of WXML component styles.
WXSS is similar to CSS in Web development. In order to be more suitable for mini program development, WXSS has been extended and modified based on CSS.

File composition

Project common style: The `app.wxss` in the root directory is the project common style and will be injected into every page of mini programs. Page style: A WXSS file with the same name and at the same level as the page registered with app.json. In the above example, `pages/rpx/index` is registered and the file `pages/rpx/index.wxss` is the style for page `pages/rpx/index.wxml` .
Other styles: Other styles can be referenced by project common styles and page styles. For referencing methods, see WXSS Reference section in this document.
Unlike web development, you don't need to optimize the number of style files requested for mini program development. You just focus on the code organization. Style files will go through compilation optimization and detailed compilation will be introduced later.

Dimension unit

In WXSS, the rpx (responsive pixel) was introduced. You can use this to adapt to different screen width to make development easier.
For example, if px is used, a page will have too much white space for the same element displayed under different screen width.

The following shows that the rpx is used:

After the mini program code is compiled, rpx will be converted to px. The conversion uses 375 physical pixels as a standard, which means 1 rpx = 1 px on a screen with 375 physical pixels width.
For example, on iPhone 6, the screen width is 375 px, and there are 750 physical pixels in total, then 1 rpx = 375 / 750 physical pixels = 0.5 px.

WXSS reference

In CSS, developers can reference another style file like this: @import url ('./ test_0.css')
This does not merge test_0.css into index.css on the request, which means that when index.css is requested, there will be an additional request for test_0.css.

In the mini program, we can still implement style reference in a way as shown below:

```
@import './test_0.wxss'
```

Since WXSS will be compiled and packaged into the object file, users only need to download it once. And there will be no redundant file requests due to style references during usage.

Inline style

WXSS inline style is consistent with that in Web development:

```
<!--index.wxml-->
<!--Inline style-->
<view style="color: red; font-size: 48rpx"></view>
```

Mini programs support dynamically updating inline styles:

```
<!--index.wxml-->

<!-- Dynamically changing inline styles-->
<!--
{
  eleColor: 'red',
  eleFontsize: '48rpx'
}
-->
<view style="color: {{eleColor}}; font-size: {{eleFontsize}}"></view>
```

Selectors

The following selectors are supported:

Type	Selector	Example	Description
class selector	.class	.intro	Selects all components with class = "intro"
ID selector	#id	#firstname	Selects the components with id = "firstname"

element selector	element	view checkbox	Selects `view` components of all documents and all `checkbox` components
pseudo-element selector	::after	view::after	Insert content after the `view` component
pseudo-element selector	::before	view::before	Insert content before the `view` component

The priority of WXSS is similar to that of CSS, with weights as shown below:

The higher the weight, the higher the priority. The style set later takes precedence over the style set earlier, given the same priority.
The priority weights of WXSS selectors:

```
view{ // Weight is 1
  color: blue
}

.ele{ // Weight is 10
  color: red
}

#ele{ // Weight is 100
  color: pink
}

view#ele{ //Weight 1 + 100 = 101, highest priority, and element color is orange
  color: orange
}

view.ele{ // Weight is 1 + 10 = 11
  color: green
}
```

Official style library

For most developers, ECMAScript and JavaScript are the same thing, but strictly speaking, they are different. ECMAScript is a standard for scripting language and it is standardized by Ecma International in the document ECMA-262. JavaScript is an implementation of ECMAScript. Knowing this helps you understand that JavaScript in mini program development is different from JavaScript in browsers and in NodeJS.

ECMA-262 specifies components of the ECMAScript language.

©2013-2025 Tencent Cloud International Pte. Ltd.

JavaScript

Last updated：2024-11-21 18:28:34

The primary programming language for mini programs is JavaScript. You can use JavaScript to develop business logic and call APIs of mini programs to fulfill business requirements.

ECMAScript

For most developers, ECMAScript and JavaScript are the same thing, but strictly speaking, they are different. ECMAScript is a standard for scripting language and it is standardized by Ecma International in the document ECMA-262. JavaScript is an implementation of ECMAScript. Knowing this helps you understand that JavaScript in mini program development is different from JavaScript in browsers and in NodeJS.
ECMA-262 specifies components of the ECMAScript language:
1. Syntax
2. Types
3. Statements
4. Keywords
5. Operators
6. Objects
JavaScript components in web browsers:

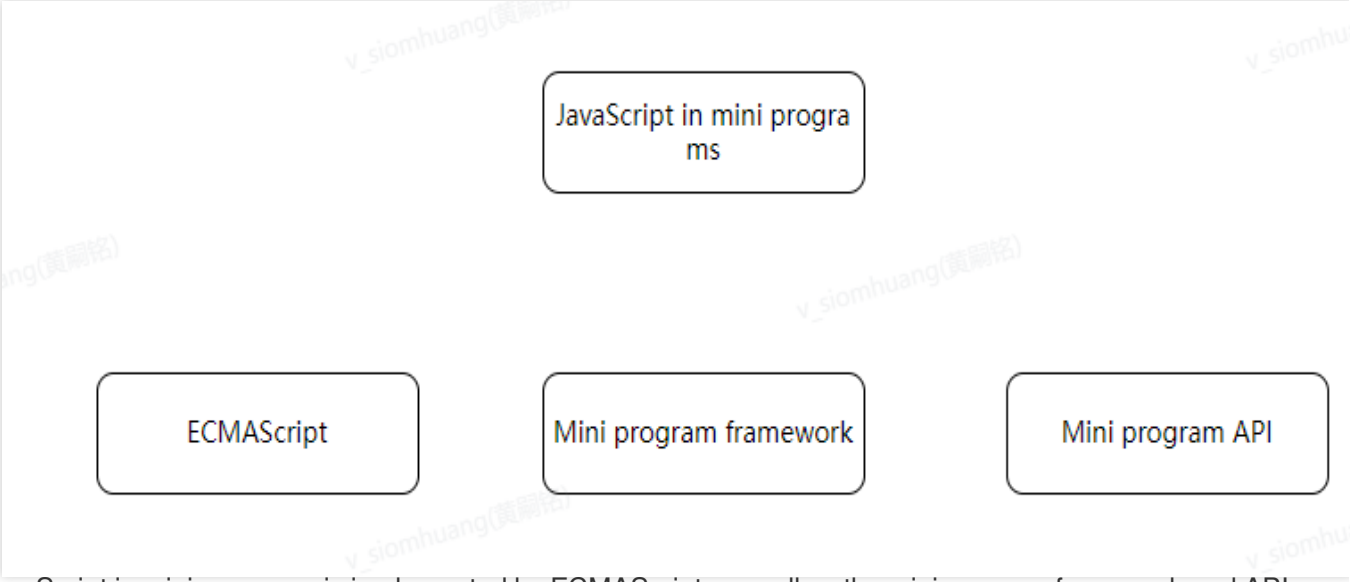


JavaScript in browsers has three important components: ECMAScript, Browser Object Model (BOM) and Document Object Model (DOM). These two object models, which are very familiar to web developers, allow developers to manipulate browser performances, such as modify URLs, modify page rendering, and record data. The JavaScript composition in NodeJS is shown in the following figure:

©2013-2025 Tencent Cloud International Pte. Ltd.



JavaScript in NodeJS is composed of ECMAScript, NPM and Native modules. NodeJS developers are very familiar with the NPM, the package manager. They can use various packages to implement features and use some native modules, such as FS, HTTP, OS, to have some capabilities that the language itself does not have.
However, different from the above mentioned environment, the mini program JavaScript compositions are as follows:

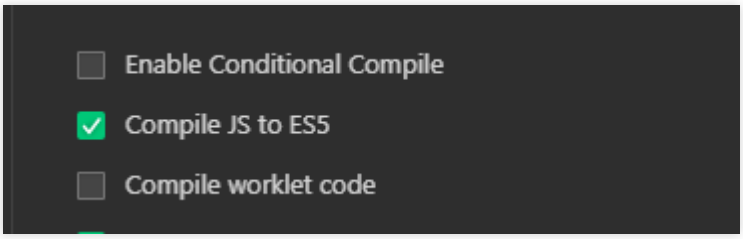


The JavaScript in mini programs is implemented by ECMAScript, as well as the mini program framework and APIs. Compared with JavaScript in browsers, there are no BOM or DOM objects, so JavaScript libraries such as JQuery and Zepto cannot run in mini programs. Similarly, due to the lack of Native modules and NPM package manager, neither native libraries can be loaded in mini programs nor most NPM packages can be directly used.

Mini program runtime environment

After understanding that JavaScript in mini programs is different from that in browsers and NodeJS, you also should be aware that the script execution environment for mini programs on different platforms is also different.
Mini programs can now run on three platforms:
iOS, including iOS9,10 and 11
Android
Mini program IDE

The ECMAScript standards are different for mini program implementation in these three platforms. Up to now, there are a total of seven versions of ECMAScript standard. Currently, most developers use ECMAScript 5 and ECMAScript 6 standards. However, for mini programs, the runtime environment used by iOS9 and iOS10 is not fully compatible with ECMAScript 6 standard. Some syntax and keywords specified in ECMAScript 6 are not available or different from the standard, such as:
Arrow functions
let const
Template strings
…
You may find that some code has syntax errors on mobile phones with earlier operating systems. In order to solve this problem, the mini program IDE provides syntax transcoding tools to help you convert ECMAScript 6 code into ECMAScript 5 code.
You need to check converting ES6 to ES5 in the project settings to use this feature.



Modularization

In browsers, JavaScript is running in the same scope, and the defined parameters or methods can be accessed or overridden by subsequently loaded scripts. Unlike browsers, mini programs can use any JavaScript file as a module and expose APIs through `module.exports` or `exports` .
For example, `B.js` references module A and uses the `multiplyBy2` method exposed by A to implement an operation multiplying a variable by 2, as shown below:

```
// moduleA.js
module.exports = function( value ){
  return value * 2;
}
```

```
// B.js

//Reference module A in B.js
var multiplyBy2 = require('./moduleA')
```

```
var result = multiplyBy2(4)
```

Use `require(path)` to import the common code in the files that need to use these modules:

```
var common = require('common.js')
Page({
  helloMINA: function() {
    common.sayHello('MINA')
  },
  goodbyeMINA: function() {
    common.sayGoodbye('MINA')
  }
})
```

Script execution order

In browsers, the scripts are executed in strict accordance with the order of loading, as shown below:

```
<html>
<head>
  <!-- a.js
  console.log('a.js')
   -->
  <script src ="a.js"></script>
  <script>
    console.log('inline script')
  </script>

  <!-- b.js
  console.log('b.js')
   -->
  <script src ="b.js"></script>
</head>
</html>
```

The output of the above code is:

```
a.js
inline script
b.js
```

The script execution order is different in mini programs. The entry point for the mini program execution is `app.js`. And the mini program execution order will follow the order defined in `require` in app.js file. For example:

```
/* a.js
```

```
console.log('a.js')
*/
var a = require('./a.js')
console.log('app.js')

/* b.js
console.log('b.js')
*/
var b = require('./b.js')
```

The output order of the above code is:

```
a.js
app.js
b.js
```

When the `app. js` is executed, the mini program will be executed in accordance with the order of pages defined by developers in the app. json, one by one, as shown below:

```
{
  "pages": [
    "pages/index/index",
    "pages/log/log",
    "pages/result/result"
  ],
  "window": {}
}
```

```
// app.js
console.log('app.js')
```

```
// pages/index/index
console.log('pages/index/index')
```

```
// pages/log/log
console.log('pages/log/log')
```

```
// pages/result/result
console.log('pages/result/result')
```

The output of above file execution:

```
app.js
pages/index/index
pages/log/log
```

```
pages/result/result
```

Scope

Unlike script files that run in a browser, mini program scripts have a scope that is more similar to NodeJS. Variables and functions declared in a JavaScript file are only valid in the file. Variables and functions with the same name can be declared in different files without affecting each other.

```
// a.js
//Define local variables
var localValue = 'a'
```

```
// b.js
// b.js cannot access the variables defined in a.js file:
//Define local variable console.log (localValue) //An error triggered: b.js cannot
```

When you need to use a global variable, you can use global function `getApp()` to obtain a global instance and set the related attribute values, setting the global variable:

```
// a.js
//Access global variable
var global = getApp()
global.globalValue = 'globalValue'
```

```
// b.js
//Access global variable
var global = getApp()
console.log(global.globalValue)
// Output globalValue
```

Please note that the example only works when the a.js is executed first and b.js is executed later. When you need to ensure that the global data can be used in any file securely, you can set in the `App()` :

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

```
// a.js
//Local variable
var localValue = 'a'
```

```
//Get global variable
var app = getApp()

//Modify global variable
app.globalData++  // The value of globalData is 2 after execution
```

```
// b.js
//Define other local variables, which will not affect the variables in a.js
var localValue = 'b'

// If a.js is executed first, the value outputted should be 2
console.log(getApp().globalData)
```

Guidance
Learn About Mini Programs
Mini Program Host Environment

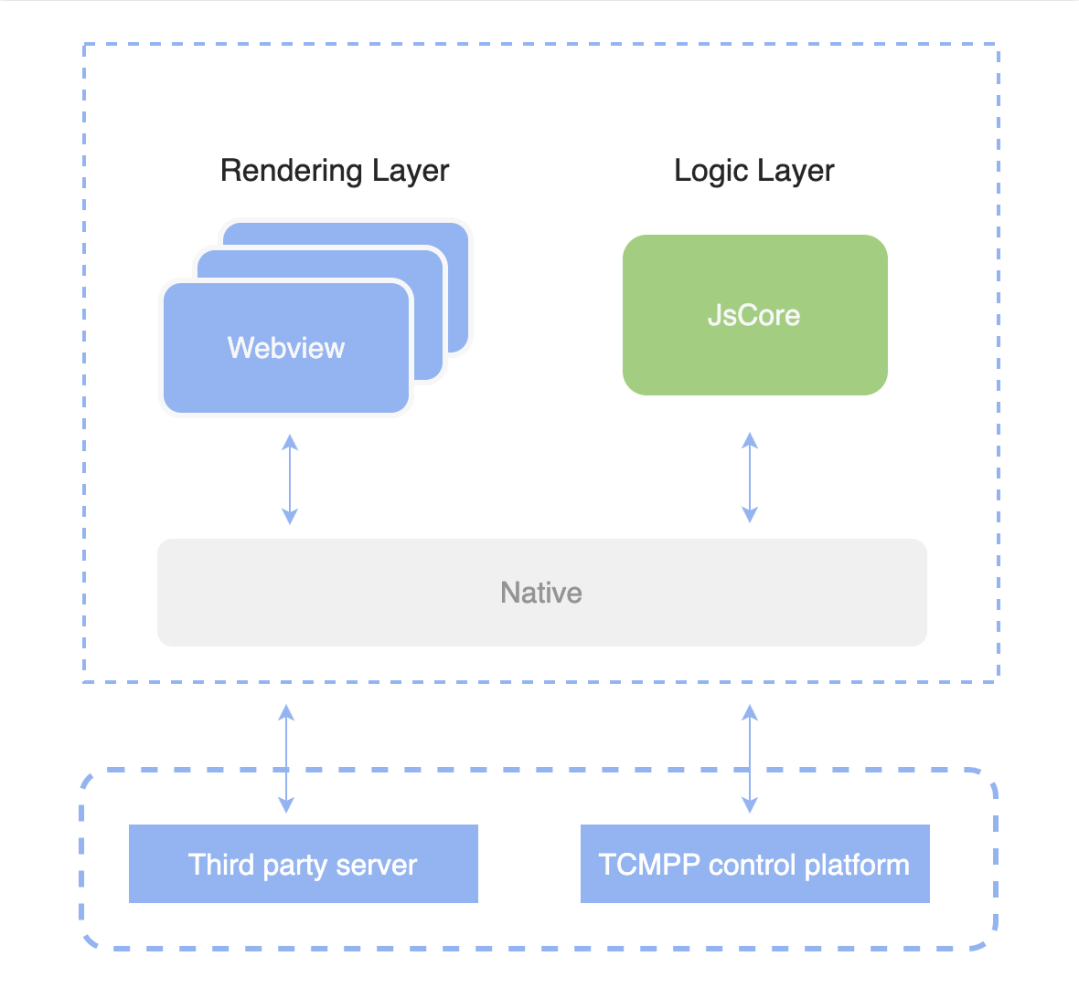
Last updated：2025-01-16 19:21:15

Our mini programs can run in all types of mobile applications. When loading and launching a mini program, the required running environment and capabilities need to be provided to the mini program by the client application. We call this set of environment " `host environment (host application)` ". With the capabilities provided by the host environment, mini programs can complete many functions that cannot be achieved by ordinary web pages.

1. Rendering Layer and Logic Layer

Firstly, let's have a brief understanding of the runtime environment of the Mini Program, which is divided into a `render layer` and a `logic layer` . WXML templates and WXSS styles are used in the render layer and JS scripts in the logic layer.
These two layers are managed via two threads: the render layer renders its interfaces using WebView threads, while the logic layer runs the JS scripts with JsCore threads. As more than one interface exists in the Mini Program, there are multiple WebView threads in the render layer. The communication between the WebView threads and JsCore threads as well as the network requests sent from the logic layer are transferred via the host app (hereinafter also referred to as Native). The communication model of the Mini Program is shown below.

©2013-2025 Tencent Cloud International Pte. Ltd.



Our mini program runs on multiple platforms: iOS/iPadOS client, Android client, and the developer tools for debugging.

The script execution environment and the environment used for component rendering are different for different running environments, and the performance varies:

On iOS, iPadOS, the JavaScript code of the logic layer of the mini program runs in JavaScriptCore, and the view layer is rendered by WKWebView, with environments such as iOS 14, iPad OS 14, and so on.

On Android, the JavaScript code of the logic layer of mini programs runs in V8, and the view layer is rendered by the self-developed XWeb engine based on Mobile Chromium kernel.

On development tools, the JavaScript code of the logic layer of the mini program runs in NW.js, and the view layer is rendered by Chromium Webview. JavaScriptCore cannot enable Just-In-Time Compiler, and the performance is significantly lower than other platforms under the same conditions.

2. Programs and Pages

The host app downloads the code package of the Mini Program to the local device before opening it.

This allows you to get all the page paths of the Mini Program from the pages field of app.json:

©2013-2025 Tencent Cloud International Pte. Ltd.

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

This configuration note defines two pages in the project, pages/index/index and pages/logs/logs, and the first page written in the pages field is the home page of the mini program (the first page you see when you open the mini program).

The path to each page can be found in the package. For example, pages/index/index, under this path, there are `WXML` , `WXSS` , `JS` files and `JSON` configuration files corresponding to this page.

After the mini program is launched, the onLaunch callback of the App instance defined in app.js is executed.

```
App({
  onLaunch() {
   // Triggered after the mini program is launched.
  }
})
```

The entire mini program has only one App instance, which is shared by all the pages, and more event callbacks refer to Registering a Mini Program Account.

Next, let's take a brief look at how a page of a mini program is written.

You can observe that under `pages/logs/logs` , there are actually 4 kinds of files. The host client will first generate an interface according to the `logs.json` configuration, and you can define the top colour and text in this json file. Immediately after that, the client loads the WXML structure and WXSS styles for the page. Finally, the client loads `logs.js` , and you can see that `logs.js` looks like this.

```
Page({
  data: { // Data involved in page rendering
    logs: []
  },
  onLoad() {
    // After the page is rendered, execute the
  }
})
```

Page is a page constructor that generates a page. When the page is generated, the Mini Program framework renders the final structure using `data` and `index.wxml` , so that the Mini Program appears as it is now.

After the interface is rendered, the page instance receives an onLoad callback, where you can process your logic.

For more information on the Page constructor, see Registering Page.

3. Components

Components are provided by mini programs for developers to create page UI and customise `WXML` . Just like HTML tags such as div, p, etc., in mini programs, you only need to write the name of the corresponding component tag in WXML to display the component on the interface. For example.

```
<view>Welcome</view>
```

The page displays a line of text: Welcome. The developer can control the style of the component by using style and class to specify the width, height, colours, and so on.

> More components can be found in the Component Overview.

4. API

In order to make it easier for developers to use the capabilities provided by mini programs, we provides a variety of APIs for developers to use, which can be used to modify the interface of the mini program, get the current location of the device, play video, audio and so on.
There are two types of APIs, synchronous and asynchronous, the synchronous APIs will return the result directly, while the asynchronous APIs will return the result through callbacks. The developer needs to process the logic according to the callback method of the API and pass in the correct parameters to process the business.
To get the user's geographic location, just call `wx.getLocation` asynchronously:

```
wx.getLocation({
  type: 'wgs84',
  success: (res) => {
    var latitude = res.latitude // latitude
    var longitude = res.longitude // longitude
  }
})
```

To retrieve device information, simply call wx.getSystemInfo synchronously or asynchronously: wx.getSystemInfo.

```
// asynchronous
wx.getSystemInfo({
  success (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)
  }
})

// synchronous
try {
  wx.getSystemInfoSync();
} catch(e) {}
```

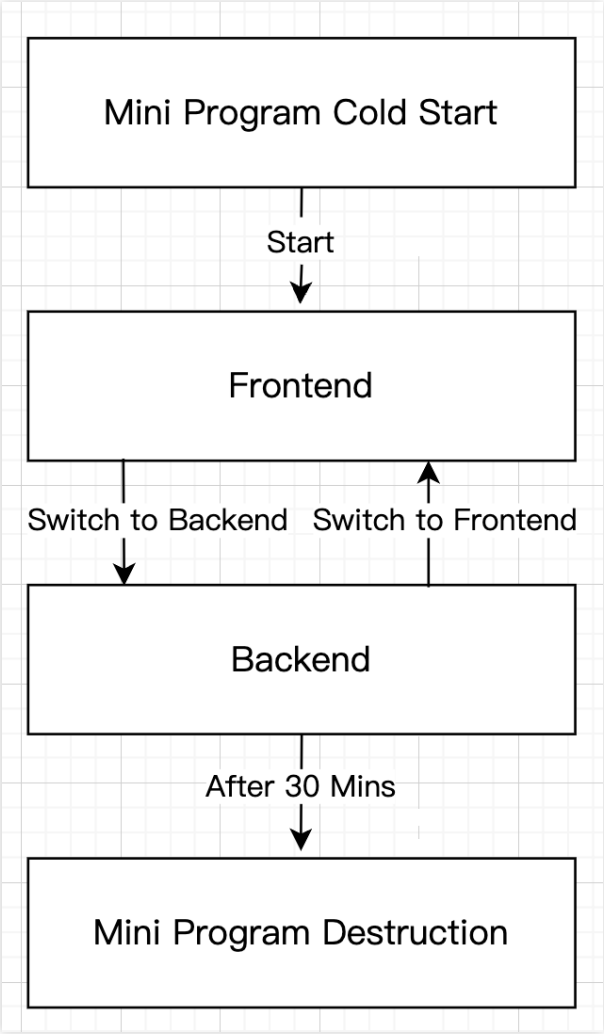
For more API capabilities see API Overview.

Operation Mechanism

Last updated：2024-11-28 10:27:55

1. Mini programme life cycle

From the time a mini program is started to the time it is finally destroyed, it will go through a variety of states and behave differently in different states.



1.1 Mini program startup

There will be two kinds of ways for mini program startup, one is cold start and the other is hot start;
Hot start: the process of switching a mini program running in the backend to the frontend when the user wants to reopen the opened mini program after a certain period of time;

Cold start: when the user opens the mini program for the first time or reopens the mini program after it has been actively destroyed by the host application, at this time, the mini program needs to be reloaded and started.

1.2 Frontend/Backend State

After the mini program is started, the user can directly see the interface of the mini program, at this time the mini program is in the frontend state.
When the user clicks the app control button in the upper right corner to "close" the mini program or presses the Home button on the mobile phone to leave the host application, the mini program is not destroyed but runs in the background, at this time, the mini program can still be run briefly for a short period of time, but the use of some APIs will be restricted.
The ways to switch to the backend include but not limited to the following:
Tap the capsule button in the upper right corner to leave the mini program;
For iOS system, swipe right from the left side of the screen to leave the mini program;
For Android system, tap the back button to leave the mini program;
Switch the host app directly to the backend when the mini program is running in the frontend (gesture or Home button);
Locking the screen directly when the mini program is running in the frontend.
When the user enters the client again and opens the mini program, the mini program will run from the backend to the frontend.

1.3 Mini program destruction mechanism

If the user has not used the mini program for a long time, or the system resources are strained, the mini program will be "destroyed", i.e. completely terminated. Specifically, it includes the following situations:
When the mini program enters the backend and is "hung", if it does not enter the frontend again for a long time (currently 30 minutes), the mini program will be destroyed.
When a mini program takes up too much system resources, it may be destroyed by the system or recycled by the host app.
On iOS system, when the host app continuously receives system memory alarms within a certain time interval, it will actively destroy the mini program according to a certain strategy and prompt the user to re-open the mini program for lack of running memory.
Specific strategies will be continuously adjusted and optimised.

2. Mini program opening logic

2.1 Open the mini program when it is cold started

When a mini program is cold started, the page of the mini program will be opened in the following cases;
If there is no path in the startup scene, it will go to the home page;

If there is the path in the startup scene, it will go to the page with the corresponding path.

2.2 Open the mini program when it is hot started

When the mini program is hot started, the page of the mini program will be opened in the following cases;

If there is no path in the startup scene, it will keep the last viewed page;

If there is the path in the startup scene, it will go to the page with the corresponding path.

©2013-2025 Tencent Cloud International Pte. Ltd.

Code Structure

Last updated：2025-03-20 18:01:55

1. JSON configuration

JSON is a data format, rather than a programming language. In the Mini Program, JSON is used for static configuration.
The following sections describes how to use `app.json` and `project.config.json` in the root directory of the project as well as `logs.json` in the `pages/logs` directory.

1.1 app.json for Mini Program Configuration

`app.json` is used for the global configuration of the Mini Program, setting all the page paths, interface behaviors, network timeouts and bottom tabs. The app.json is configured in the QuickStart project as below:

```
{
  "pages":[
    "pages/index/index",
    "pages/logs/logs"
  ],
  "window":{
    "backgroundTextStyle":"light",
    "navigationBarBackgroundColor": "#fff",
    "navigationBarTitleText": "TCSAS",
    "navigationBarTextStyle":"black"
  }
}
```

Meaning of each configuration item:
The pages field - describes all the page paths of the Mini Program, so that the Weixin app knows the directory where your Mini Program page is defined.
The window field - defines the background color and text color at the top of all pages in the Mini Program.

For more information on other configuration items, see app.json for Mini Program Configuration.

1.2 project.config.json for Tool Configuration

Usually when you use a tool, you do some personalised configurations for your own preferences, such as compilation configurations, and you have to reconfigure them when you change to another computer and reinstall the tool.

©2013-2025 Tencent Cloud International Pte. Ltd.

With this in mind, the mini program developer tool generates a project.config.json in the root directory of each project, and any configurations you make on the developer tool are written to this file. When you reinstall the tool or change computers to work on it, you can just load the code package of the same project, and the developer tool will automatically help you restore the personalised configurations to the ones that were used at the time when you were developing the project, which will include the name of the project, the name of the project, and the name of the developer. This will include the name of the project, the option to automatically compress the code when uploading it, and a number of other options.

1.3 page.json for Page Configuration

`page.json` is used for page-related configuration of the Mini Program, for example, the logs.json under the `pages/logs` directory.

If you want the entire Mini Program to be blue, you can specify the top color as blue in `app.json` . However, this may not be the case in reality. You may want to use different colors for different pages to indicate different function modules. Therefore, `page.json` is provided to allow developers to define the properties of each page separately, such as the top color, the option to enable "Pull Down to Refresh", etc.

For more information on other configuration items, see Page Configuration.

1.4 JSON Syntax

Note the following regarding JSON configuration in the Mini Program.

JSON files are wrapped in curly brackets ({}), containing data in the form of key-value pairs. JSON keys must be enclosed in double quotes.

Note：

1. Values in JSON files must be in one of the following data formats. Any other format will trigger an error, such as "undefined" in JavaScript:

Numbers, including floating-point numbers and integers;

Strings enclosed in double quotes;

Booleans with a value of true or false;

Arrays enclosed in square brackets ([]);

Objects enclosed in curly brackets ({});

Null.

2. Also note that JSON files do not support comments. Adding comments will trigger an error.

2. WXML Template

All web page programmers know that web pages are built using `HTML` , `CSS` and `JavaScript` , where HTML describes the structure of the page, CSS determines the appearance of the page, and JS defines the interaction between the page and the user.

Similar roles exist in the Mini Program, where `WXML` is the equivalent of `HTML`. By opening `pages/index/index.wxml`, you will see the following:

```
<view class="container">
  <view class="userinfo">
    <button wx:if="{{!hasUserInfo && canIUse}}"> getUserInfo </button>
    <block wx:else>
      <image src="{{userInfo.avatarUrl}}" background-size="cover"></image>
      <text class="userinfo-nickname">{{userInfo.nickName}}</text>
    </block>
  </view>
  <view class="usermotto">
    <text class="user-motto">{{motto}}</text>
  </view>
</view>
```

Similar to HTML, WXML consists of tags and properties. However, there are also some differences, shown as below:

1. Tag names are different

div, p and span are three most commonly used tags when writing HTML, which can be combined in different ways to create different components, such as calendars and pop-ups. Since these common components are required in almost all scenarios, we can put them into packages to improve development efficiency.

In the above example, the WXML of the Mini Program uses view, button and text tags. In addition to these native basic capabilities, we also provide packaged component capabilities like map, video and audio.

2. Properties like wx:if and expressions like are added

When developing web pages, we generally manipulate DOM (the tree generated from the description of HTML) with JS to allow the interface to respond to users' behaviors. For example, when a user clicks a button, JS records the states in JS variables and manipulates DOM properties or behaviors via the DOM API to trigger changes in the interface. However, as the project gets more and more complex, your code will be full of interface interaction logic and program state variables, which is not conducive to development. Therefore, the MVVM developer mode (e.g., React, Vue) is created to separate rendering from logic. To put it simply, instead of manipulating DOM directly, JS only needs to manage the states, and describes the relationship between states and interface structure via another template syntax.

This also applies to the framework of the Mini Program. If you want to display the Hello World string in the interface:

Write the WXML as follows:

```
<text>{{msg}}</text>
```

Use JS to manage the state only:

```
this.setData({ msg: "Hello World" })
```

The process that binds a variable to the interface via the syntax is called data binding. Data binding is not enough to describe the relationship between states and interface. Control capabilities such as if/else and for are required, which

are expressed with properties that start with wx: in the Mini Program.

> Refer to WXML Syntax Reference for more details.

3. WXSS Style

WXSS has most of the features of CSS, but incorporates some new features and modifications for the Mini Program.
1. New size unit is added. When writing the CSS style, developers need to convert the pixel units to adapt to different mobile device screens with different widths and pixel ratios. WXSS supports the new rpx unit at its underlying layer, allowing the Mini Program to take over the job from developers and convert the units at the underlying layer. Since the units are converted using floating-point operations, the result may deviate slightly from the expected one.
2. Global and local styles are provided. Similar to the above app.json and page.json, app.wxss can be written as the global style, which applies to all the pages of the Mini Program. The local page style page.wxss only applies to the current page.
3. WXSS only supports certain CSS selectors
Refer to WXSS for more details.

4. JS Logic Interaction

It's not enough for a service to just display the interface. Interaction with users is required, such as responding to user's clicks and obtaining user's location. In the Mini Program, we process user's operations by writing JS scripts.

```
<view>{{ msg }}</view>
<button bindtap="clickMe">Click me</button>
```

When a button is clicked, we want to display msg as "Hello World" on the interface. To do this, we declare the bindtap property on the button and the clickMe method in the JS file to respond to this click:

```
Page({
  clickMe: function() {
    this.setData({ msg: "Hello World" })
  }
})
```

Responding to user actions is as simple as that. For more details on events, see View Layer - Event System.

Directory Structure

Last updated：2024-11-21 18:34:18

The mini program comprises an overarching `app` that defines the entire application, along with multiple `page` elements, each describing their respective pages.

1. The core of a mini program is constituted by three essential files, which must be situated in the root directory of the project, as follows:

File	Indispensable	Effect
app.js	Yes	Mini program logic
app.json	Yes	Mini program's public configuration
app.wxss	No	List of mini program's public styles

2. A mini program page is composed of four files, namely:

File	Indispensable	Effect
js	Yes	Page logic
wxml	Yes	Page Structure
json	No	Page Configuration
wxss	No	Page style list

Note:

To facilitate developers and reduce configuration items, the four files describing the page must possess identical paths and filenames.

Files allowed to be uploaded

Within the project directory, the following files undergo compilation and hence cannot be directly accessed after upload: *.js, app.json, *.wxml, *.wxss (wherein wxml and wxss files are only applicable to pages configured in app.json). Beyond this, only files with suffixes included in the allowlist can be uploaded. The specific allowlist is as follows:

1. qs
2. png

3. jpg
4. jpeg
5. gif
6. svg
7. json
8. cer
9. mp3
10. aac
11. m4a
12. mp4
13. wav
14. ogg
15. silk

©2013-2025 Tencent Cloud International Pte. Ltd.

Configuring Mini Program Global and Page Configuration

Last updated：2024-11-21 18:34:18

1. Global Configuration

The `app.json` file in the root directory of the mini program is used to configure the global configuration of the mini program, deciding the path of the page file, the window performance, setting the network timeout, setting multiple tabs and so on.

The following is an app.json file that contains some common configuration options:

```
{
  "pages": [
    "pages/index/index",
    "pages/logs/index"
  ],
  "window": {
    "navigationBarTitleText": "Demo"
  },
  "tabBar": {
    "list": [{
      "pagePath": "pages/index/index",
      "text": "index"
    }, {
      "pagePath": "pages/logs/index",
      "text": "log"
    }]
  },
  "debug": true
}
```

Please refer to global configuration for a complete description of the configuration items.

2. Page Configuration

Each mini program page can also use the same name .json file to configure the window of the page, the configuration items in the page will override the same configuration items in the window of app.json.

For example:

```
{
  "navigationBarBackgroundColor": "#ffffff",
  "navigationBarTextStyle": "black",
  "navigationBarTitleText": "title",
  "backgroundColor": "#eeeeee",
  "backgroundTextStyle": "light"
}
```

Please refer to Page Configuration for full configuration item descriptions.

Mini Program Framework Introduce

Last updated：2025-03-10 16:12:24

Mini program development framework is designed to allow developers to develop services with a native app experience in Weixin by the simplest and most efficient means possible.
The Mini Program framework system includes Logic Layer (App Service) and View Layer (View). The Mini Program provides its own view layer description languages WXML and WXSS, and a logic layer framework based on JavaScript, as well as a data transfer and event system between the view layer and the logic layer, allowing developers to focus on data and logic.

Reactive Data Binding

The reactive data binding system is the core of the framework. It provides an extremely simple way to ensure the synchronization of data and views. To modify data, you simply need to modify the data on the logic layer and it will be updated in the view layer.
Here is a simple example:

```
<!-- This is our View -->
<view> Hello {{name}}! </view>
<button bindtap="changeName"> Click me! </button>
```

```
// This is our App Service.
// This is our data.
var helloData = {
  name: 'TCSAS'
}

// Register a Page.
Page({
  data: helloData,
  changeName: function(e) {
    // sent data change to view
    this.setData({
      name: 'World'
    })
  }
})
```

The developer uses the framework to bind `name` in the logic layer data to `name` in the view layer, so `Hello TCSAS!` appears once the page opens.
When the button is clicked, the view layer sends `changeName` events to the logic layer, which finds and executes the corresponding event handler.
After the callback function is triggered, the logic layer executes `setData` to change the name in `data` from TCSAS to `World`. Since the data is bound to the view layer, `Hello World!` is automatically applied to the view layer.

Page Management

The `framework` manages the page routing of the entire Mini Program, which enables seamless switching between pages and gives the pages a complete lifecycle. The developer only needs to register the data, methods, and lifecycle functions of the page to the `framework`. All other complex operations are handled by the `framework`.

Basic Components

The framework provides a set of basic components that come with unified styles and special logic. Developers can create powerful mini programs by combined use of these components.

Wide Range of APIs

The `framework` features a wide range of Weixin native APIs that can easily call the capabilities provided by Weixin, such as access to user information, local storage, and payment.

©2013-2025 Tencent Cloud International Pte. Ltd.

Logic Layer

Last updated：2025-03-20 18:01:55

1. Introduce

At the logic layer of the mini program development framework, the `JavaScript` engine is used to provide mini programs with features specific to mini programs and an environment for running JavaScript code written by developers.
The logic layer processes data and sends processed data to the view layer, and receives event feedback from the view layer.
All code written by developers will be packaged into a `JavaScript` file, which runs when the Mini Program is started until the Mini Program is destroyed. This behavior is similar to ServiceWorker. Therefore, the logic layer is also referred to as an `App Service` .
Based on `JavaScript` , we have added some features to facilitate the development of Mini Programs:
App and Page methods used for Mini Program Registration and Registering Page;
getApp and getCurrentPages methods, respectively used to obtain the App instance and the current page stack;
Rich API, such as getting geolocation, swiping and other unique capabilities.;
Modularization capability, allowing each page to have an independent scope.
Note：
The logic layer of the Mini Program development framework does not run on browsers. Therefore, web-based capabilities of JavaScript, such as window and document, are unavailable.

2. Registering a Mini Program Account

For each Mini Program, a Mini Program instance needs to be registered by calling the App method via app.js, to link the lifecycle callback function, error listening function, page not found listening function, and the like.
For specific definitions and usage of the parameters, refer to App.

```
// app.js
App({
  onLaunch (options) {
    // Do something initial when launch.
  },
  onShow (options) {
    // Do something when show.
  },
  onHide () {
```

```
    // Do something when hide.
  },
  onError (msg) {
    console.log(msg)
  },
  globalData: 'I am global data'
})
```

A Mini Program has only one app instance, which is shared to all pages. Developers can obtain the globally unique app instance by using the `getApp` method, and then obtain data in the app or call a function registered with App by the developers.

```
// xxx.js
const appInstance = getApp()
console.log(appInstance.globalData) // I am global data
```

3. Registering Page

For each page of a Mini Program, a page instance needs to be registered by calling the Page method via the js file corresponding to the page, to specify the initial data of the page, the lifecycle callback function, the event processing function, and the like of the page.

3.1 Registering a Page with the Page Constructor

Simple pages can be constructed using Page().

Code example:

```
// index.js
Page({
  data: {
    text: "This is page data."
  },
  onLoad: function(options) {
    // Do some initialize when page load.
  },
  onReady: function() {
    // Do something when page ready.
  },
  onShow: function() {
    // Do something when page show.
  },
  onHide: function() {
    // Do something when page hide.
  },
```

```
  onUnload: function() {
    // Do something when page close.
  },
  onPullDownRefresh: function() {
    // Do something when pull down.
  },
  onReachBottom: function() {
    // Do something when page reach bottom.
  },
  onShareAppMessage: function () {
    // return custom share data when user share.
  },
  onPageScroll: function() {
    // Do something when page scroll
  },
  onResize: function() {
    // Do something when page resize
  },
  onTabItemTap(item) {
    console.log(item.index)
    console.log(item.pagePath)
    console.log(item.text)
  },
  // Event handler.
  viewTap: function() {
    this.setData({
      text: 'Set some data for updating view.'
    }, function() {
      // this is setData callback
    })
  },
  customData: {
    hi: 'MINA'
  }
})
```

For specific definitions and usage of the parameters, refer to Page.

3.2 Constructing Pages with the Component Constructor

The Page constructor works for simple pages. However, for complex pages, the Page constructor may not work well. In this case, you can use the Component constructor to construct the page. The main difference of the Component constructor is that the methods need to be inside methods: { }.

Code example:

```
Component({
  data: {
    text: "This is page data."
  },
  methods: {
    onLoad: function(options) {
      // Executed when the page is created
    }, onPullDownRefresh: function() { // Execute on page creation.
    onPullDownRefresh: function() {
      // onPullDownRefresh: function() { // On page creation }, onPullDownRefresh:
    }, onPullDownRefresh.
    // Event response function
    viewTap: function() {
      // ...
    }
  }
})
```

This is very similar to a Custom Components in that you can use advanced features such as behaviors in the same way as a custom component.

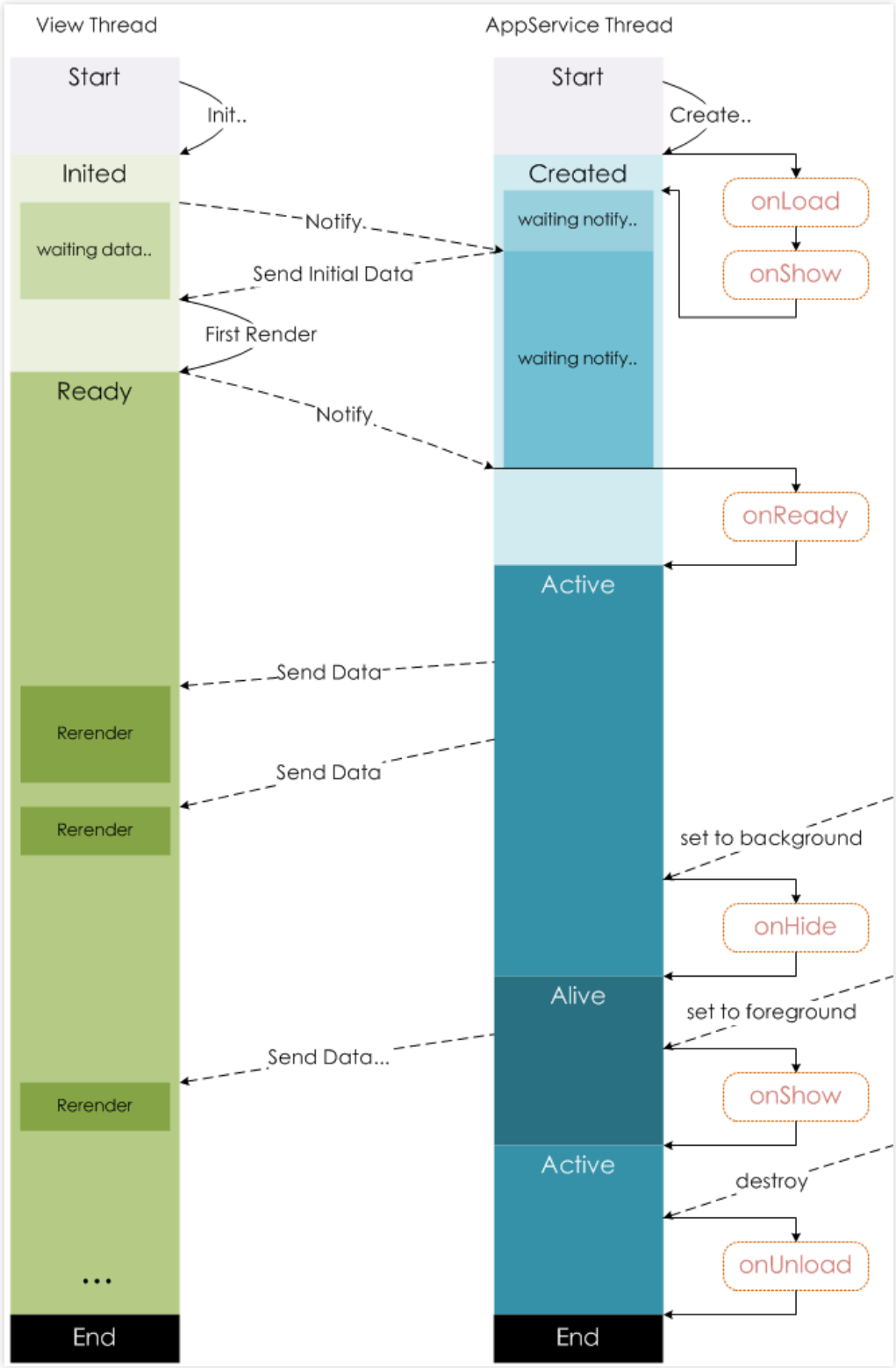
> Read the Component Constructors section for more details.

4. Page Lifecycle

Note

You don't need to fully understand the following content right away, but it will help in the future.

The following figure describes the lifecycle of a `Page` instance.



5. Page Routing

The framework manages all the routes to pages of a Mini Program.

5.1 Page Stack

The framework maintains all the existing pages via stacks. When a route is switched, the page stack reacts as follows:

Routing	Reaction on the Page Stack
Initialization	The new page is pushed onto the stack.
Opening a new page	The new page is pushed onto the stack.
Page redirection	The current page is popped out of the stack, and the new page is pushed onto the stack.
Page return	Pages are successively popped out of the stack until the destination return page is displayed.
Tab switching	All the pages are popped out of the stack, and only the new tab page is retained.
Reloading	All the pages are popped out of the stack, and only the new page is retained.

Developers can use the getCurrentPages() function to get the current page stack, given as an array in the order of the stack, with the first element being the home page and the last element being the current page.

Note：

Don't try to modify the page stack, it will cause routing and page state errors.

Don't call getCurrentPages() on App.onLaunch when the page has not been generated.

5.2 Routing

The following shows how page routing is triggered and the lifecycle functions for pages.

Routing	Trigger Condition	Source Page	Routed-To Page
Initialization	A page of the Mini Program is opened for the first time.	-	onLoad, onShow
Opening a new page	The API wx.navigateTo is called. The component <navigator open-type="navigateTo"/> is used.	onHide	onLoad, onShow
Page redirection	The API wx.redirectTo is called. The component <navigator open-type="redirectTo"/> is used.	onUnload	onLoad, onShow
Page return	The API wx.navigateBack is called. The component <navigator open-type="navigateBack"> is used. The user taps the return button on the top left.	onUnload	onShow

Tab switching	The API wx.switchTab is called. The component <navigator open-type="switchTab"/> is used. The user switches between tabs.	-	For details, refer to the following table.
Restart	The API wx.reLaunch is called. The component <navigator open-type="reLaunch"/> is used.	onUnload	onLoad, onShow

Lifecycle functions corresponding to tab switching (in the following example, pages A and B are Tabbar pages, page C is obtained after page A is opened, and page D is obtained after page C is opened):

Current Page	Routed-To Page	Triggered Lifecycle Function (In Order)
A	A	Nothing happend
A	B	A.onHide(), B.onLoad(), B.onShow()
A	B (Opened again)	A.onHide(), B.onShow()
C	A	C.onUnload(), A.onShow()
C	B	C.onUnload(), B.onLoad(), B.onShow()
D	B	D.onUnload(), C.onUnload(), B.onLoad(), B.onShow()
D (Entered due to redirection)	A	D.onUnload(), A.onLoad(), A.onShow()
D (Entered due to redirection)	B	D.onUnload(), B.onLoad(), B.onShow()

Note：

navigateTo and redirectTo can be used to open non-tabBar pages only.

switchTab can be used to open tabBar pages only.

reLaunch can be used to open any pages.

The tabBar on the bottom of a page is determined by the page. In other words, tabBar is displayed on the bottom of all pages defined as tabBar.

Parameters used to call page routing can be obtained via onLoad in the destination page.

6. Modules

6.1 Modularization

Some common codes can be extracted to create an independent js file as a module. APIs for the module can be exposed only via module.exports or exports.

Note：

exports is a reference to module.exports. Therefore, a random modification to the point of exports in the module may cause an unknown error. Developers are advised to expose module APIs via module.exports if they do not know the relationship between them.

Currently, node_modules cannot be directly passed into Mini Programs. To use `node_modules`, developers are advised to copy relevant code to the directory of the Mini Program, or use the `npm` feature supported by Mini Programs.

```
// common.js
function sayHello(name) {
  console.log(`Hello ${name} !`)
}
function sayGoodbye(name) {
  console.log(`Goodbye ${name} !`)
}

module.exports.sayHello = sayHello
exports.sayGoodbye = sayGoodbye
```

To pass common code into files that need to access these modules, use require.

```
var common = require('common.js')
Page({
  helloMINA: function() {
    common.sayHello('MINA')
  },
  goodbyeMINA: function() {
    common.sayGoodbye('MINA')
  }
})
```

6.2 File Scope

Variables and functions declared in the JavaScript file are valid only in the file. Variables and functions with the same names can be declared in different files without affecting each other.

The global application instance can be obtained via the global function `getApp`. To obtain the global data, you can set `App()`. For example:

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

```
// a.js
// The localValue can only be used in file a.js.
var localValue = 'a'
// Get the app instance.
var app = getApp()
// Get the global data and change it.
app.globalData++
```

```
// a.js
// The localValue can only be used in file a.js.
var localValue = 'a'
// Get the app instance.
var app = getApp()
// Get the global data and change it.
app.globalData++
```

Note：

It's worth noting that require needs to use **relative paths** when introducing modules.

7. API

Mini Program development framework to provide rich micro-channel native API, you can easily tune up WeChat provides capabilities, such as access to user information, local storage, payment functions. Please refer to API Overview。

Usually, in Mini Programs API There are several types:

7.1 Event monitor API

We made a pact to on Initial API Used to listen if an event is triggered, such as:wx.onSocketOpen, wx.onCompassChange Wait.

this kind API Accepts a callback function as a parameter that is called when an event is triggered, and passes in the relevant data as parameters.

Code examples:

```
wx.onCompassChange(function (res) {
  console.log(res.direction)
})
```

7.2 synchronization API

We made a pact to Sync Ending API It's all in sync. API， Such as wx.setStorageSync， wx.getSystemInfoSync Wait. In addition, there are some other synchronization API, such as wx.createWorker， wx.getBackgroundAudioManager For details, see API Instructions in the documentation.

synchronization API Can be obtained directly from the return value of the function, and an exception can be thrown if the execution goes wrong.

Code examples

```
try {
  wx.setStorageSync('key', 'value')
} catch (e) {
  console.error(e)
}
```

7.3 asynchronous API

Majority API It's all asynchronous. API, such as wx.request， wx.login Wait. this kind API Interfaces usually accept a Object Type, which supports specifying the following fields on demand to receive interface call results.

Object Parameter specification

Parameter name	type	Required	Introductions
success	function	no	Interface calls the successful callback function
fail	function	no	Interface calls failed callback functions
complete	function	no	Callback function at the end of an interface call (both successful and unsuccessful calls are executed)
Other	Any	-	Additional parameters defined by the interface

Parameters to the callback function

success， fail， complete When the function is called, it passes in a Object Type parameter with the following fields:

attribute	type	Introductions
errMsg	string	Error message if the call returns successfully ${apiName}:ok
errCode	number	Error code, partial only API Support, please refer to the corresponding meaning API Documentation, if successful 0。
Other	Any	Interface to return additional data

Code examples

```
wx.login({
  success(res) {
    console.log(res.code)
  }
})
```

View Layer

Last updated：2025-03-20 18:01:55

The view layer of the framework is written using WXML and WXSS and presented by components.
WXML (WeiXin Markup language) is used to describe the page structure;
WXS (WeiXin Script) is a scripting language for Mini Programs. Combined with WXML, it can be used to construct the page structure;
WXSS (WeiXin Style Sheet) is used to describe the page style.
Components are the building blocks of a view.

1. WXML

WXML (WeiXin Markup Language) is a markup language for framework design. It can be used to build the page structure when combined with base components and event system.
For more information on WXML syntax, see WXML syntax.
The following examples show what WXML can do:

1.1 Data Binding

```
<!-- wxml -->
<view> {{message}} </view>
```

```
// page.js
Page({
  data: {
    message: 'Hello MINA!'
  }
})
```

1.2 List Rendering

```
<!--wxml-->
<view wx:for="{{array}}"> {{item}} </view>
```

1.3 Condition Rendering

```
<!--wxml-->
<view wx:if="{{view == 'WEBVIEW'}}"> WEBVIEW </view>
```

```
<view wx:elif="{{view == 'APP'}}"> APP </view>
<view wx:else="{{view == 'MINA'}}"> MINA </view>
```

```
// page.js
Page({
  data: {
    view: 'MINA'
  }
})
```

1.4 Template

```
<!--wxml-->
<template name="staffName">
  <view>
    FirstName: {{firstName}}, LastName: {{lastName}}
  </view>
</template>

<template is="staffName" data="{{...staffA}}"></template>
<template is="staffName" data="{{...staffB}}"></template>
<template is="staffName" data="{{...staffC}}"></template>
```

```
// page.js
Page({
  data: {
    staffA: {firstName: 'Hulk', lastName: 'Hu'},
    staffB: {firstName: 'Shang', lastName: 'You'},
    staffC: {firstName: 'Gideon', lastName: 'Lin'}
  }
})
```

2. WXSS

WXSS (WeiXin Style Sheets) is a set of style languages that describe WXML component styles, and is used to determine how WXML components are displayed.

WXSS has most of the features of CSS to cater for the needs of front-end developers, but incorporates some new features and modifications to adapt to the development of Weixin Mini Programs.

Compared with CSS, WXSS offers the following additional features:

Dimension unit

Style import;

Dimension Unit.

Note：

For iOS 8 and below, if you use flexbox layout, you need to add `display: -webkit-flex` property.

2.1 Dimension Unit

rpx (responsive pixel): Adaptable to the screen width. The specified screen width is 750 rpx. If the screen width on iPhone6 is 375 px (750 physical pixels), then 750 rpx = 375 px = 750 physical pixels, i.e. 1 rpx = 0.5 px = 1 physical pixel.

Device	rpx-to-px conversion (screen width/750)	px-to-rpx conversion (750/screen width)
iPhone5	1rpx = 0.42px	1px = 2.34rpx
iPhone6	1rpx = 0.5px	1px = 2rpx
iPhone6 Plus	1rpx = 0.552px	1px = 1.81rpx

Note：

Tip: Designers can use iPhone6 as a standard in Visual Design when developing Weixin Mini Programs.

Note: It is inevitable to have some glitches on small screens. Please try to avoid this during development.

2.2 Style Import

You can use the `@import` statement to import external style sheet. The `@import` statement should be followed by the relative path of the external style sheet and ends with ;.

```
/** common.wxss **/
.small-p {
  padding:5px;
}
```

```
/** app.wxss **/
@import "common.wxss";
.middle-p {
  padding:15px;
}
```

2.3 Inline Style

You can use "style" and "class" properties for framework components to control the component style.

style: Static styles are all written to "class". "style" receives dynamic styles and parses them at runtime. It is not recommended to write static styles to "style" to avoid affecting the rendering speed.

```
/** common.wxss **/
```

```
.small-p {
  padding:5px;
}
```

class: Used to specify style rules. Its value is a collection of class selector names (style class names) in the style rules. The style class names do not contain . and are separated with spaces.

```
<view class="normal_view" />
```

2.4 Selectors

The following selectors are supported:

Selector	Example	Description
.class	.intro	Selects all components with class="intro"
#id	#firstname	Selects the component with id="firstname"
element	view	Selects all "view" components
element, element	view, checkbox	Selects all "checkbox" components and the "view" components of all documents
::after	view::after	Inserts content after the "view" component
::before	view::before	Inserts content before the "view" component

2.5 Global and Local Styles

A global style is defined in app.wxss and applies to every page, while a local one is defined in the page's wxss file and only applies to the current page (overrides the same selector in app.wxss).

3. WXS

WXS (WeiXin Script) is a script segment inlined in WXML. WXS can be used to inline a small number of processing scripts in the template to enrich the data preprocessing capability of the template. In addition, WXS can also be used to write simple WXS event response functions.

Syntactically, WXS is similar to JavaScript with a few limitations. For a complete overview of the WXS syntax, see WXS - Introduction.

Here are some simple examples of using WXS.

Page Rendering

```
<!--wxml-->
```

```
<wxs module="m1">
var msg = "hello world";

module.exports.message = msg;
</wxs>

<view> {{m1.message}} </view>
```

Page output:

```
hello world
```

Data Processing

```
// page.js
Page({
  data: {
    array: [1, 2, 3, 4, 5, 1, 2, 3, 4]
  }
})
```

```
<!--wxml-->
<!-- The following getMax function takes an array and returns the value of the larg
<wxs module="m1">
var getMax = function(array) {
  var max = undefined;
  for (var i = 0; i < array.length; ++i) {
    max = max === undefined ?
      array[i] :
      (max >= array[i] ? max : array[i]);
  }
  return max;
}

module.exports.getMax = getMax;
</wxs>

<!-- Calls the getMax function in WXS with the "array" in page.js as the parameter
<view> {{m1.getMax(array)}} </view>
```

Page output:

```
5
```

4. Event System

4.1 What Is an Event?

An event is a channel through which the view layer communicants with the logical layer;

An event can pass user's action on to the logical layer for processing;

An event can be bound to a component. When the event is triggered, the corresponding event handler is executed at the logic layer;

An event object can have additional information such as ID, dataset, touches.

4.2 How to Use an Event

Bind an event handler to the component.

For example, if the event is bindtap, the event handler can be found in the page when a user taps the component.

```
<view id="tapTest" data-hi="Weixin" bindtap="tapName"> Click me! </view>
```

Write the event handler in the Page definition with " `event` " as the parameter.

```
Page({
  tapName: function(event) {
    console.log(event)
  }
})
```

The log content is as follows:

```
{
  "type":"tap",
  "timeStamp":895,
  "target": {
    "id": "tapTest",
    "dataset":  {
      "hi":"Weixin"
    }
  },
  "currentTarget":  {
    "id": "tapTest",
    "dataset": {
      "hi":"Weixin"
    }
  },
  "detail": {
    "x":53,
    "y":14
  },
```

```
  "touches":[{
    "identifier":0,
    "pageX":53,
    "pageY":14,
    "clientX":53,
    "clientY":14
  }],
  "changedTouches":[{
    "identifier":0,
    "pageX":53,
    "pageY":14,
    "clientX":53,
    "clientY":14
  }]
}
```

4.3 Respond to Events with WXS Function

4.3.1 Background

The effect of frequent user interactions on mini programs is laggy, for example, if there are two elements A and B on a page, and the user makes a touchmove gesture on A, and asks B to follow the gesture, movable-view is a typical example. The response to a touchmove event is as follows:
The touchmove event is thrown from the rendering layer (Webview) to the logical layer (App Service);
The logical layer (App Service) handles the touchmove event and then changes the position of B through setData.
A touchmove response requires two communications between the logic layer and the rendering layer and one rendering, which is time-consuming. In addition, the setData rendering will also block the execution of other scripts, resulting in a delay in the animation process of the whole user interaction.

4.3.2 Solution

The basic idea of this solution is to reduce the number of communications and let the events respond in the rendering layer (Webview). The framework of the mini program is divided into the rendering layer (Webview) and the logic layer (App Service), the purpose of this layering is to control, the developer's code can only run in the logic layer (App Service), and this idea must allow the developer's code to run in the rendering layer (Webview).
WXS functions used to respond to mini program events, can only respond to the built-in component events, does not support custom component events.
WXS function in addition to the pure logic of the operation, but also through the encapsulated ComponentDescriptor instance to access and set the component class and style, for interactive animation, set the style and class enough.
WXS function accepts two parameters, the first is event, in the original event based on the `event.instance` object, the second parameter is `ownerInstance` , and `event.instance` is a `ComponentDescriptor` object. The specific use is as follows:
WXS functions for binding and registering event handling in a component

```
<wxs module="wxs" src="./test.wxs"></wxs>
<view id="tapTest" data-hi="Weixin" bindtap="{{wxs.tapName}}"> Click me! </view>
**Note: Bound WXS functions must be enclosed in {{}}**
```

The test.wxs file implements the tapName function.

```
function tapName(event, ownerInstance) {
  console.log('tap Weixin', JSON.stringify(event))
}
module.exports = {
  tapName: tapName
}
```

ownerInstance contains methods to set the style and class of the component.

An example of a WXS function is shown below:=

```
var wxsFunction = function(event, ownerInstance) {
  var instance = ownerInstance.selectComponent('.classSelector') // Returns an inst
  instance.setStyle({
      "font-size": "14px" // Support for rpx
  })
  instance.getDataset()
  instance.setClass(className)
  // ...
  return false // does not bubble up, which is equivalent to calling both stopPropa
}
```

The input parameter `event` is the ComponentDescriptor instance of the component that triggered the event, which is the ComponentDescriptor instance of the component that triggered the event, and the ownerInstance is the instance of the page if the component that triggered the event is inside the page.

The ownerInstance represents the `ComponentDescriptor` instance of the component that triggered the event. If the component that triggered the event is inside a page, the ownerInstance represents the page instance.

The `ComponentDescriptor` currently supports the following APIs.

Method	Parameter	Description
selectComponent	selector objects	Returns the `ComponentDescriptor` instance of the component
selectAllComponents	Array of selector objects	Returns an array of `ComponentDescriptor` instances of the component
setStyle	Object/string	Set the style of the component, supports rpx, the style set has higher priority than the style

		defined inside the component wxml. Cannot set the style of the topmost page
addClass/removeClass/hasClass	string	Set the class of the component, the class priority is higher than the class defined inside the component wxml. Cannot set the class of the top level page
getDataset	-	Return the dataset of the current component/page
callMethod	(funcName:string, args:object)	Call the function defined in the logic layer (App Service) of the current component/page. funcName is the name of the function, args is the parameter of the function
getState	-	Returns an object, use this method when there are local variables that need to be stored for subsequent use
triggerEvent	(eventName, detail)	Same as the triggerEvent of the component

4.3.3 Usage

WXML defines the event:

```
<wxs module="test" src="./test.wxs"></wxs>
<view
  change:prop="{{test.propObserver}}"
  prop="{{propValue}}"
  bindtouchmove="{{test.touchmove}}"
  class="movable">
</view>
```

The change:prop above (with the change: prefix in front of the property) triggers the WXS function when the prop property is set, and the value must be enclosed in `{{}}` . Similarly the observer property inside properties defined by Component is triggered after a `setData({propValue: newValue})` call.

Note：

WXS functions must be enclosed in `{{}}` . The WXS function is triggered when the value of the prop is set, not just when the value is changed, so the WxsPropObserver function is called once when the page is initialised.

The WXS file `test.wxs` defines and exports event handlers and functions triggered by property changes:

```
module.exports = {
  touchmove: function(event, instance) {
      console.log('log event', JSON.stringify(event))
  },
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
  propObserver: function(newValue, oldValue, ownerInstance, instance) {
      console.log('prop observer', newValue, oldValue)
  }
}
```

4.4Events

4.4.1 Event Types

Events are divided into bubbling events and non-bubbling ones.

Bubbling event: When an event is triggered on a component, this event is propagated to the component's parent node;

Non-bubbling event: When an event is triggered on a component, this event is not propagated to the component's parent node.

WXML bubbling events:

Type	Trigger Condition
touchstart	Finger touch starts
touchmove	Finger moves after touch
touchcancel	Finger touch is interrupted by call reminder, pop-up window, etc.
touchend	Finger touch ends
tap	The finger leaves the screen after touch
longpress	The finger leaves the screen after it taps and holds on the screen for more than 350 ms. If an event callback function is specified and this event is triggered, the tap event is not triggered.
longtap	The finger leaves the screen after it taps and holds on the screen for more than 350 ms (it is recommended to use longpress event instead).
transitionend	Triggered when a WXSS transition or wx.createAnimation animation ends
animationstart	Triggered when a WXSS animation starts
animationiteration	Triggered after an iteration of a WXSS animation ends
animationend	Triggered when a WXSS animation completes
touchforcechange	Triggered when the screen is tapped again on an iPhone supporting 3D Touch.

Note：

Unless otherwise specified, the custom events of components other than those listed above are non-bubbling events, such as the submit event of form, input event of input, and scroll event of scroll-view. For more information, see Component Overview.

4.4.2 Event Binding and Bubbling

Event bindings are written similarly to component properties, e.g.:

```
<view bindtap="handleTap">
    Click here!
</view>
```

If the user clicks on this view, the page's handleTap is called.
The event binding function can be a data binding, e.g.:

```
<view bindtap="{{ handlerName }}">
  Click here!
</view>
```

In this case, the page's this.data.handlerName must be a string specifying the name of the event handler; if it's an empty string, the binding is disabled (you can use this feature to temporarily disable some events).
In most components and custom components, bind can be immediately followed by a colon, and the meaning remains the same, e.g. bind:tap

4.4.3 Binding and stopping event bubbling

In addition to bind, events can also be bound with catch. Unlike bind, catch prevents events from bubbling up.
For example, in the following example, clicking on the inner view calls handleTap3 and handleTap2 successively (because the tap event bubbles up to the middle view, which prevents the tap event from bubbling up to the parent node), and clicking on the middle view triggers handleTap2, which triggers handleTap2, which triggers handleTap2, which triggers handleTap2, which triggers handleTap2. Click outer view will trigger handleTap1.

```
<view id="outer" bindtap="handleTap1">
  outer view
  <view id="middle" catchtap="handleTap2">
    middle view
    <view id="inner" bindtap="handleTap3">
      inner view
    </view>
  </view>
</view>
```

4.4.4 Mutually exclusive event binding

As of Base Library version 1.5.44, you can use mut-bind to bind events in addition to bind and catch. When a mut-bind is triggered, if the event bubbles to another node, the mut-bind bindings on the other node will not be triggered, but the bind and catch bindings will still be triggered.
In other words, all mut-binds are "mutually exclusive" and only one of them will be triggered. At the same time, it doesn't affect the binding effect of bind and catch at all.
For example, in the example below, clicking inner view will call handleTap3 and handleTap2, and clicking middle view will call handleTap2 and handleTap1.

```
<view id="outer" mut-bind:tap="handleTap1">
  outer view
  <view id="middle" bindtap="handleTap2">
    middle view
    <view id="inner" mut-bind:tap="handleTap3">
      inner view
    </view>
  </view>
</view>
```

4.4.5 Event Capturing

touch-related events support capturing. In event capturing, which precedes bubbling, events arrive at nodes in an order reverse of that in which they do during bubbling. To listen to events during capturing, use `capture-bind` or `capture-catch` as the keyword. The latter interrupts the capturing and cancels bubbling.
In the code below, tap "inner view" to call `handleTap2` , `handleTap4` , `handleTap3` , and `handleTap1` in turn.

```
<view id="outer" bind:touchstart="handleTap1" capture-bind:touchstart="handleTap2">
  outer view
  <view id="inner" bind:touchstart="handleTap3" capture-bind:touchstart="handleTap4
    inner view
  </view>
</view>
```

If the first capture-bind in the above code is changed to capture-catch, only handleTap2 is triggered.

```
<view id="outer" bind:touchstart="handleTap1" capture-catch:touchstart="handleTap2"
  outer view
  <view id="inner" bind:touchstart="handleTap3" capture-bind:touchstart="handleTap4
    inner view
  </view>
</view>
```

4.4.6 Event Objects

Unless otherwise specified, when the component triggers an event, the handler bound to the event at logic layer receives an event object.

BaseEvent object properties:

Property	Type	Description
type	String	Event Type
timeStamp	Integer	Timestamp when the event is generated
target	Object	A collection of property valuesfor the component triggering the event
currentTarget	Object	A collection of property valuesfor the current component

CustomEvent object properties (inherit from BaseEvent):

Property	Type	Description
detail	Object	Additional information

TouchEvent object properties (inherit from BaseEvent):

Property	Type	Description
touches	Array	Touch event. An array of information of touch points staying in the screen.
changedTouches	Array	Touch event. An array of information of touch points involving changes in the screen

Special events: The touch events in canvas are non-bubbling events and thus have no currentTarget.

4.4.7 type

Represents the type of an event.

4.4.8 timeStamp

Number of milliseconds from the moment when the page is opened until the event is triggered.

4.4.9 target

The source component triggering the event.

Property	Type	Description
id	String	ID of the event source component
dataset	Object	A collection of custom properties starting with data- on the event source component

4.4.10 currentTarget

The current component bound to the event.

Property	Type	Description
id	String	Current component ID
dataset	Object	A collection of custom properties starting with data- on the current component

Note：

You can refer to the above example for "target" and "currentTarget". When you tap "inner view", the event objects "target" and "currentTarget" received by handleTap3 are both "inner", but the "target" and "currentTarget" received by handleTap2 are "inner" and "middle", respectively.

4.4.11 dataset

You can add some custom data to the component node so as to obtain the custom node data from the event for logical processing.

In WXML, these custom data begins with data- and multiple words are joined with hyphen -. In the code, a hyphenation is converted to camel case, and the uppercase characters are automatically converted to lowercase ones. For example:

data-element-type is converted to event.currentTarget.dataset.elementType;

data-elementType is converted to event.currentTarget.dataset.elementtype.

Example:

```
<view data-alpha-beta="1" data-alphaBeta="2" bindtap="bindViewTap"> DataSet Test </
```

```
Page({
  bindViewTap:function(event){
    event.currentTarget.dataset.alphaBeta === 1 // - will convert to camel case
    event.currentTarget.dataset.alphaBeta === 2 // uppercase will be converted to l
  }
})
```

4.4.12 touches

"touches" is an array, in which each element is a Touch object ("touches" in the canvas touch event is a CanvasTouch array). Touch points staying in the screen.

Touch objects

Property	Type	Description
identifier	Number	Touch point identifier

pageX, pageY	Number	Distance from the upper left corner of the document. The top left corner is the origin, the horizontal line is X-axis, and the vertical line is Y-axis.
clientX, clientY	Number	Distance from the upper left corner of the page display area (the area on the screen without navigation pane). The horizontal line is X-axis and the vertical line is Y-axis.

CanvasTouch objects

Property	Type	Description
identifier	Number	Touch point identifier
x, y	Number	Distance from the upper left corner of Canvas. Canvas's upper left corner is the origin, the horizontal line is X-axis, and the vertical line is Y-axis.

4.4.13 changedTouches

Indicates the touch points involving changes (in the same format as "touches"), including touchstart (appearance of touch point), touchmove (movement of touch point), and touchend/touchcancel (disappearance of touch point).

4.4.14 detail

The data carried by the custom event. For example, the submission event of a form component carries user's input, and the error event of media carries the error message. For more information, see the event definitions in Component Overview.

Similar to pageX and page Y, the x and y in detail of the tap event represent the distance from the upper left corner of the document.

5. Basic Components

The framework provides developers with a set of basic components that can be combined to achieve quick development. For more information, see Component Overview.

What is component?

A component is the basic building block of a view layer;

A component includes a `start tag` , `end tag` , property (modifies the component), and content (between the two tags).

```
<tagname property="value">
Content goes here ...
</tagname>
```

Note：

All components and properties are in lowercase and joined with a hyphen -

Property Types

Type	Description	Comment
Boolean	Boolean value	If a component has this property, the property value is always considered true regardless of what the actual value is. The property value is false only when the component has no this property. If the property value is a variable, the variable is converted to a Boolean value.
Number	Number	1, 2.5
String	String	"string"
Array	Array	[1, "string"]
Object	Object	{ key: value }
EventHandler	Event handler	"handlerName" is the event handler name defined in Page
Any	Any property	-

Common Properties

All components have the following properties:

Property	Type	Description	Comment
id	String	A unique identifier of a component	Remains unique across the entire page.
class	String	Component style class	The style class defined in the WXSS
style	String	Component inline style	This is an inline style that can be dynamically set
hidden	Boolean	Whether to display the component	All components are displayed by default
data-*	Any	Custom property	When an event is triggered on the component, it is sent to the event handler
bind* / catch*	EventHandler	Component event	See Event

Special Properties

Almost all components have their own custom properties. You can modify the features or style of the component. See the definition of each Component Overview.

6. Getting Node Information on Interface

6.1 WXML Node Information

The API for querying node information is used to get information of a node such as its properties, style, and position in the interface.

This API is mostly used to query the current position and scroll position of a node in the interface.

Sample code:

```
const query = wx.createSelectorQuery()
query.select('#the-id').boundingClientRect(function(res){
  res.top // Coordinates of the upper boundary of the #the-id node (relative to the
})
query.selectViewport().scrollOffset(function(res){
  res.scrollTop // the vertical scroll position of the display area
})
query.exec()
```

In the above example, #the-id is a node selector, which is slightly different from the CSS selector. For more information, see SelectorQuery.select.

In custom components or pages that contain custom components, it is recommended to use this.createSelectorQuery, instead of wx.createSelectorQuery, to ensure nodes are selected within the correct area.

6.2 WXML Node Layout Intersection

The APIs for node layout intersection is used to listen to the layout intersection between two or more component nodes. This set of APIs are often used to determine whether certain nodes are visible to users and the percentage of the visible area.

The set of APIs involve the following concepts:

Reference node: The reference node for listening, whose layout area is taken as the reference area. If there are multiple reference nodes, the intersection of their layout areas is taken as the reference area. The page display area can also be used as a reference area;

Target node: The target to be listened to. Only one target node is allowed by default (you can listen to multiple nodes at the same time using the selectAll option);

Intersection area: The intersection area between the layout area of the target node and the reference area;

Intersection percentage: The percentage of the intersection area in the reference area

Threshold: If the intersection percentage reaches the threshold, the callback function of the listener is triggered. Multiple thresholds can be set;

©2013-2025 Tencent Cloud International Pte. Ltd.

In the following sample code, a callback function is triggered each time the target node (specified using the selector .target-class) enters or exits the page display area.

Sample code:

```
Page({
  onLoad: function(){
    wx.createIntersectionObserver().relativeToViewport().observe('.target-class', (
      res.id // target node id
      res.dataset // target node dataset
      res.intersectionRatio // the proportion of the intersection area to the layou
      res.intersectionRect // The intersection region
      res.intersectionRect.left // the coordinates of the left boundary of the inte
      res.intersectionRect.top // the upper boundary of the intersection region
      res.intersectionRect.width // width of the intersection region
      res.intersectionRect.height // height of the intersection region
    })
  }
})
```

The following sample code can be in the target node (specified with the selector .target-class) and the reference node (specified with the selector .relative-class) in the page display area of the intersection or separation, and the degree of intersection or separation to the extent of the layout of the target node of the area of the 20% and 50%, trigger the callback function.

Sample code:

```
Page({
  onLoad: function(){
    wx.createIntersectionObserver(this, {
      thresholds: [0.2, 0.5]
    }).relativeTo('.relative-class').relativeToViewport().observe('.target-class',
      res.intersectionRatio // the proportion of the intersection area to the layou
      res.intersectionRect // The intersection region
      res.intersectionRect.left // the coordinates of the left boundary of the inte
      res.intersectionRect.top // the upper boundary of the intersection region
      res.intersectionRect.width // width of the intersection region
      res.intersectionRect.height // height of the intersection region
    })
  }
})
```

Note:

The intersection area with the page display area does not accurately represent the area visible to users, because the area involved in the calculation is the "layout area", which may be cropped or hidden by, for example, a node whose ancestor node has the overflow style "hidden", or obscured by, for example, a node with fixed position during drawing.

©2013-2025 Tencent Cloud International Pte. Ltd.

In custom components or pages that contain custom components, it is recommended to use this.createIntersectionObserver, instead of wx.createIntersectionObserver, to ensure nodes are selected within the correct area.

7. Respond to Changes in Display Area

7.1 Display Area Size

The display area refers to the area in Mini Program interface where you can arrange elements freely. By default, the size of the display area remains unchanged after the page initialization. But you can change the default size by either of the following two ways.

Enable Screen Rotation on a Phone

To allow the Mini Program pages to support screen rotation, set "pageOrientation": "auto" in the `window` segment of `app.json`, or configure `"pageOrientation": "auto"` in the page json file.

The following example shows how to enable screen rotation in a single page json file.

Code example:

```
{
"pageOrientation": "auto"
}
```

If the above declaration is added to the page, the page rotates and the display area is resized as the screen rotates. `pageOrientation` can be set to landscape, which locks the screen orientation to "Landscape".

Enable Screen Rotation on an iPad

To allow the Mini Program to support screen rotation, add "resizable": true to app.json.

Code example:

```
{
"resizable": true
}
```

If the above declaration is added to the Mini Program, the Mini Program page rotates and the display area is resized as the screen rotates.

Note：

You cannot enable or disable screen rotation for a single page.

7.2 Media Query

Sometimes, the page layout varies with the size of the display area. You can use "media query" to solve most of the problems.

Code example:

```
.my-class {
  width: 40px;
}

@media (min-width: 480px) {
  /* Style rules that only work on screens 480px or wider */
  .my-class {
    width: 200px;
  }
}
```

7.3 Screen Rotation Event

Sometimes, subtle layout changes cannot be controlled by using "media query" alone. In this case, js can be used.

You can use selectorQuery.selectViewport to read the size of the display area of the page in js.

The onResize of the page can be used to listen to the page resizing event. For custom components, you can use the "resize lifecycle" function to listen to the event. The callback function returns the size of the display area.

Code example:

```
Page({
  onResize(res) {
    res.size.windowWidth // The width of the new display area.
    res.size.windowHeight // the height of the new display area
  }
})
```

```
Component({
  pageLifetimes: {
    resize(res) {
    res.size.windowWidth // The width of the new display area.
    res.size.windowHeight // the height of the new display area
    }
  }
})
```

In addition, you can also use wx.onWindowResize to listen to the event (not recommended).

8. Animation

8.1 Common User Interface Animations

In Mini Programs, you can use CSS transitions and CSS animations to create simple UI animations.

During the animation creation, you can use `bindtransitionend`, `bindanimationstart`, `bindanimationiteration`, and `bindanimationend` to listen to animation events.

Event	Description
transitionend	A CSS transition ends or a step of wx.createAnimation ends
animationstart	CSS animation starts
animationiteration	A step of CSS animation ends
animationend	A CSS animation ends

Note：
These events are not bubbling events and need to be bound to the node that is being animated to take effect.
It is also possible to use the wx.createAnimation interface to dynamically create simple animation effects (the keyframe animation interface described below is recommended in the new version of the mini program base library instead).

8.2 Advanced Animations

The above animations may not be applicable to complicated scenarios.
You can dynamically adjust the "style" property of a node by responding to events with WXS. Changing the value of "style" continuously can produce animation effects. And this method can be used to dynamically generate animations based on users' touch events.
Animation effects can also be achieved by using setData to change the user interface continuously. This can change the interface arbitrarily, but usually produces a large delay or stutter, and even causes the Mini Program to freeze. In this case, you can improve performance by changing the setData of the page to the setData in Custom Components.
The following example shows how to generate a stopwatch animation using setData.

Custom Components

Last updated：2025-03-25 18:15:55

1. Introduce

Developers can abstract feature modules in a page into custom components to reuse them in different pages, or split a page into multiple low-coupling modules for easy code maintenance. Custom components are used in almost the same way as base components.

1.1 Creating Custom Component

Similar to a page, a custom component consists of `json`, `wxml`, `wxss`, and `js` files. To write a custom component, you first need to make a custom component declaration in the json file (set the files as custom `components` by setting the component field to `true`):

```
{
  "component": true
}
```

In the meanwhile, write a component template in the wxml file and a component style in the wxss file in a same way that you write a page. For details and considerations, see Component Template and Style.

Code example:

```
<!-- This is the internal WXML structure of the custom component -->
<view class="inner">
  {{innerText}}
</view>
<slot></slot>
```

```
/*  This style only applies to this custom component  */
.inner {
  color: red;
}
```

Note：

ID selectors, attribute selectors, and tag name selectors should not be used in the component's wxss.

In the custom component's js file, you need to register the component using `Component()` and provide the component's property definitions, internal data, and custom methods.

The property values and internal data of the component are used to render the component's `wxml`. The property values are passed in from outside the component. For details, see Component Constructor.

©2013-2025 Tencent Cloud International Pte. Ltd.

Code example:

```
Component({
  properties: {
    // The innerText property is defined here, and the property value can be specif
    innerText: {
      type: String,
      value: 'default value',
    }
  },
  data: {
    // Here are some component internal data
    someData: {}
  },
  methods: {
    // Here is a custom method
    customMethod: function(){}
  }
})
```

1.2 Using Custom Component

Before using a registered custom component, make a reference declaration in the json file of the page. You need to provide the tag name of each custom component and the file path of the corresponding custom component:

```
{
  "usingComponents": {
    "component-tag-name": "path/to/the/custom/component"
  }
}
```

As such, you can use custom components like you would with base components in the `wxml` of the page. The node name is the tag name of the custom component, and the node property is the property value passed to the component.

Note：

The usingComponents field declared in app.json is considered to be a global custom component and can be used directly in pages or custom components within the mini program without having to declare it again.

Code example:

```
<view>
  <!-- The following is a reference to a custom component -->
  <component-tag-name inner-text="Some text"></component-tag-name>
</view>
```

The custom component's `wxml` node structure, after being combined with the data, will be inserted into the reference location.

©2013-2025 Tencent Cloud International Pte. Ltd.

1.3 Notes

Please note the following:

Since the tag name of a WXML node can only be a combination of **lowercase letters**, **hyphens (-)**, and **underscores (_)**, the tag name of a custom component can contain these characters only.

Custom components can also reference other custom components in a similar way to referencing custom components by pages (the usingComponents field is used).

The name of the project's root directory where the custom components and pages are located cannot be prefixed with "wx-". Otherwise, an error will occur.

Note that using `usingComponents` in the page file will make the prototype of the page's `this` object slightly different, including:

The prototype that uses the `usingComponents` page differs from the prototype that does not use `usingComponents`, i.e. the result of `Object.getPrototypeOf(this)` is different.

More methods, such selectComponent, are available when `usingComponents` is used.

For performance reasons, when usingComponents is used, the content of setData is not deep copied directly, that is, this.data.field === obj after this.setData({ field: obj }) is called. (Deep copy occurs when this value is passed between components.)

For complicated pages, retest is required if the usingComponents definition field is added or removed.

2. Component Template and Style

Similar to a page, each custom component has its own `wxml` template and `wxss` style.

2.1 Component Template

Component templates are written in the same way as page templates. The node tree generated by binding the component template with the component data will be inserted into the location where the component is referenced.

A `<slot>` node can be provided in the component template to host the child nodes provided via component reference.

Code example:

```
<!-- Component template -->
<view class="wrapper">
  <view>This indicates the internal node of the component</view>
  <slot></slot>
</view>
```

```
<!--Page template where the component is referenced-->
<view>
  <component-tag-name>
```

```
    <!-- This part is added to the location of the  <slot>  component -->
    <view>This is inserted in the slot component</view>
  </component-tag-name>
</view>
```

Note：

the custom component referenced in the template and the name of its corresponding node should be explicitly defined in the json file, otherwise the node will be deemed as meaningless. Besides, the node name can also be declared as an Abstract Node.

2.2 Template Data Binding

Like an ordinary WXML template, data binding can also be employed to pass dynamic data to child component properties.

Code example:

```
<!--Page template where the component is referenced-->
<view>
  <component-tag-name prop-a="{{dataFieldA}}" prop-b="{{dataFieldB}}">
    <!-- This part is added to the location of the  <slot>  component -->
    <view>This is inserted in the slot component</view>
  </component-tag-name>
</view>
```

In the above example, the component properties `propA` and `propB` receive data passed from the page, and the page can change linked data field via `setData` .

Note：

Only JSON compatible data can be passed via the above data binding.

2.3 slot of Component's WXML

The `slot` node can be included in the component's wxml to host the wxml structure provided by the component user.

By default, a component's wxml can contain only one `slot` . Multiple slots can be enabled by declaring in the component js if needed.

```
Component({
  options: {
    multipleSlots: true // Enable multiple slots in the options of component defini
  },
  properties: { /* ... */ },
  methods: { /* ... */ }
})
```

In this case, multiple `slots` can be used in the wxml of this component, each with a unique `name` .

```
<!-- Component template -->
<view class="wrapper">
  <slot name="before"></slot>
  <view>This indicates the internal details of the component</view>
  <slot name="after"></slot>
</view>
```

You can use the slot property to insert nodes into different slots.

```
<!--Page template where the component is referenced-->
<view>
  <component-tag-name>
    <!-- This part is added to the location of the  <slot name="before">  component
    <view slot="before">This is inserted in the slot name="before" component</view>
    <!-- This part is added to the location of the  <slot name="after">  component
    <view slot="after">This is inserted in the slot name="after" component</view>
  </component-tag-name>
</view>
```

2.4 Component Style

The style of a component for the wxss file is only valid for the nodes in the component's wxml. Note the following when writing a component style:

For the components and the pages that reference the components, the class selector, instead of the id selector (`#a`), property selector (`[a]`) and tag name selector, should be used;

Unexpected behaviors may occur with the descendant selector (`.a .b`) used in the components and the pages that reference the components in extreme cases. If so, avoid using it;

The child selector (`.a>.b`) can be only used for the view component and its child nodes; unexpected behaviors may occur if it is used in other components;

Inheritance styles, such as `font` and `color` , are inherited into the components from outside the components;

In addition to inheritance styles, the styles in `app.wxss` and the styles of the pages that reference the components are invalid for custom components (unless the component style isolation option is changed).

```
#a { } /* Cannot be used in components */
[a] { } /* Cannot be used in components */
button { } /* Cannot be used in components */
.a > .b { } /* Invalid unless  .a  is  view  component node */
```

In addition, a default style can be specified for the node where the component resides by using the `:host` selector

Code example:

```
/* Component custom-component.wxss */
:host {
  color: yellow;
```

```
}
```

```
<!-- page's WXML -->
<custom-component>The text here is highlighted in yellow</custom-component>
```

2.5 Component Style Isolation

Sometimes a component wants to accept incoming style classes. In this case, several external style classes can be defined in the Component using the externalClasses definition segment.

This feature can be used to implement a hover-class property similar to that of the view component: the page can provide a style class that gives the view its hover-class, which is itself written in the page rather than in the view component's implementation.

Note：

When using a normal style class and an external style class on the same node, the priority of both classes is undefined, so it is best to avoid this situation.

Code example:

```
/* Component custom-component.js */
Component({
  externalClasses: ['my-class']
})
```

```
<!-- Component  custom-component.wxml -->
<custom-component class="my-class">The colour of this text is determined by the cla
```

In this way, the user of the component can specify the class corresponding to this style class, just like with normal properties.

Code example:

```
<!-- page's WXML -->
<custom-component my-class="red-text" />
<custom-component my-class="large-text" />
```

```
.red-text {
  color: red;
}
.large-text {
  font-size: 1.5em;
}
```

2.6 Making components accept global styles

By default, the styling of a custom component is only affected by the custom component wxss. Except in the following two cases:
The app.wxss or the page's wxss uses tag name selectors (or some other special selectors) to directly specify styles that affect the page and the component in its entirety. This is generally not a recommended practice.
Activating the addGlobalClass option on a specific custom component makes it possible for that custom component to be affected by all style definitions in app.wxss or the page's wxss.
To activate the addGlobalClass option, simply set the options.addGlobalClass field to true in the Component constructor.

Note：

When the `addGlobalClass` option is activated, there is a risk of external styles contaminating the component style, so please choose carefully.

Code example:

```
/* Component  custom-component.js */
Component({
  options: {
    addGlobalClass: true,
  }
}
```

```
<!-- Component  custom-component.wxml -->
<text class="red-text">
  The colour of this text is determined by the style definitions in `app.wxss` and
</text>
```

```
/* app.wxss */
.red-text {
  color: red;
}
```

3. Component Constructor

The Component constructor is used to define components. You can specify component properties, data, and methods when calling the Custom Components.

```
Component({

  behaviors: [],

  properties: {
```

```
    myProperty: { // Property name
      type: String,
      value: ''
    },
    myProperty2: String // Simplified definition
  },

  data: {}, // Private data, used for template rendering

  lifetimes: {
    // Lifecycle function, which can be a function or a method name defined in the
    attached: function () { },
    moved: function () { },
    detached: function () { },
  },

  // Lifecycle function, which can be a function or a method name defined in the me
  attached: function () { }, // The declaration in the attached field is overwritte
  ready: function() { },

  pageLifetimes: {
    // Component page lifecycle function
    show: function () { },
    hide: function () { },
    resize: function () { },
  },

  methods: {
    onMyButtonTap: function(){
      this.setData({
        // The properties and data are updated in the same way as with the page dat
      })
    },
    // The internal method starts with an underscore
    _myPrivateMethod: function(){
      // Here, data.A[0].B is set to 'myPrivateData'
      this.setData({
        'A[0].B': 'myPrivateData'
      })
    },
    _propertyChange: function(newVal, oldVal) {

    }
  }

})
```

In the properties definition segment, the property name is written in camel case (propertyName); in wxml, the corresponding property value is written in hyphenated case (component-tag-name property-name="attr value") when specifying the property value, and in camel case when applying it to data binding (attr=" {{propertyName}}").

3.1 Construct a Page via Component Constructor

In fact, the Mini Program's page can also be considered as a custom component. Thus, its page can also be constructed using the Component constructor, as it has the same definition field and instance method as a general component. However, you must include the usingComponents definition field in the corresponding json file.
In this case, the properties of the component can be used to receive page parameters, such as visited pages /pages/index/index?paramA=123&paramB=xyz. If the property paramA or paramB is declared, they will be assigned 123 or xyz.
The page's lifecycle method (i.e., the method starting with on) must be written in the methods definition field.
Code example:

```
{
  "usingComponents": {}
}
```

```
Component({
  properties: {
    paramA: Number,
    paramB: String,
  },

  methods: {
    onLoad: function() {
      this.data.paramA // Value of page parameter paramA
      this.data.paramB // Value of page parameter paramB
    }
  }
})
```

One of the benefits of using the Component constructor to construct pages is that you can use behaviors to extract code segments that are common to all pages.
For example, if you want to execute the same piece of code when all pages are created and destroyed, you can extract that code into behaviors.
Code example:

```
// page-common-behavior.js
module.exports = Behavior({
  attached: function() {
    // On page creation
    console.info('Page loaded!')
```

```
  },
  detached: function() {
    // When the page is destroyed
    console.info('Page unloaded!')
  }
})
```

```
// page A
var pageCommonBehavior = require('./page-common-behavior')
Component({
  behaviors: [pageCommonBehavior],
  data: { /* ... */ },
  methods: { /* ... */ },
})
```

```
// page B
var pageCommonBehavior = require('./page-common-behavior')
Component({
  behaviors: [pageCommonBehavior],
  data: { /* ... */ },
  methods: { /* ... */ },
})
```

4. Inter-component Communication and Events

4.1 Inter-component Communication

The following methods are used for communication between components.

WXML data binding: This is used by a parent component to set data for the specified property of a child component. Only JSON-compatible data is supported. For more information, see Component Template and Style.

Event: This is used by a child component to send any type of data to a parent component.

If the two methods above cannot meet your needs, the parent component can also get the instance objects of the child component via the this.selectComponent method. This allows you to directly access any component data or method.

4.2 Monitoring Events

The event system is one of the methods of communication between components. Custom components can trigger any events, and pages that references the components can monitor these events. For the basic concept and usage of events, see Events.

Customer component events are monitored in the same way as with basic component events:

Code example:

```
<!-- The "onMyEvent" method is called when the custom component triggers "myevent"
<component-tag-name bindmyevent="onMyEvent" />
<!-- Or write it like this -->
<component-tag-name bind:myevent="onMyEvent" />
```

```
Page({
  onMyEvent: function(e){
    e.detail // The detail object provided when the custom component triggers an ev
  }
})
```

4.3 Triggering Event

When a custom component triggers an event, the `triggerEvent` method should be used to specify the event name, detail object, and event options:

Code example:

```
<!-- In the custom component -->
<button bindtap="onTap">Click this button to trigger the "myevent" event</button>
```

```
Component({
  properties: {},
  methods: {
    onTap: function(){
      var myEventDetail = {} // detail object, provided to the event monitoring fun
      var myEventOption = {} // Event triggering options
      this.triggerEvent('myevent', myEventDetail, myEventOption)
    }
  }
})
```

Event triggering options include:

Option	Type	Required	Default	Description
bubbles	Boolean	No	false	Indicates whether the event is a bubbling event
composed	Boolean	No	false	Indicates whether the event can cross component boundaries. If it is false, the event is only triggered in the node tree of the referenced component and will not enter other components.
capturePhase	Boolean	No	false	Indicates whether the event has a capture phase

For more information on bubbling and capture phase, see Events.

Code example:

```
// page page.wxml
<another-component bindcustomevent="pageEventListener1">
  <my-component bindcustomevent="pageEventListener2"></my-component>
</another-component>
```

```
// Component  another-component.wxml
<view bindcustomevent="anotherEventListener">
  <slot />
</view>
```

```
// Component  my-component.wxml
<view bindcustomevent="myEventListener">
  <slot />
</view>
```

```
// Component  my-component.js
Component({
  methods: {
    onTap: function(){
      this.triggerEvent('customevent', {}) // Only pageEventListener2 is triggered
      this.triggerEvent('customevent', {}, { bubbles: true }) // pageEventListener2
      this.triggerEvent('customevent', {}, { bubbles: true, composed: true }) // pa
    }
  }
})
```

4.4 Getting component instances

This.selectComponent can be called from the parent component to get the instance of the child component.

You need to pass a match selector, e.g. `this.selectComponent(".my-component").`

The detailed syntax of selector can be found in the selector.

Code example:

```
// parent component
Page({
  data: {},
  getChildComponent: function () {
    const child = this.selectComponent('.my-component');
    console.log(child)
  }
})
```

©2013-2025 Tencent Cloud International Pte. Ltd.

In the above example, the parent component will get the instance object of the child component with class my-component, i.e. this of the child component.

If you want to customise the data returned by selectComponent, you can use the built-in behavior: wx://component-export

When this behaviour is used, the export definition field in the custom component will be used to specify the return value of the component when it is called by selectComponent.

Code example:

```
// Custom Components my-component Internal
Component({
  behaviors: ['wx://component-export'],
  export() {
    return { myField: 'myValue' }
  }
})
```

```
<! -- When using custom components -->
<my-component id="the-id" />
```

```
// Called by the parent component
const child = this.selectComponent('#the-id') // be tantamount to { myField: 'myVal
```

In the above example, the parent component gets the object `{ myField: 'myValue' }` when it gets the child component instance with `id` `the-id` .

5. Component Lifecycle

Component lifecycles refer to the functions in the component itself. These functions are automatically triggered at certain timepoints or when specific framework events occur.

The most important lifecycles are `created` , `attached` , and `detached` , which include the main time points in the entire lifecycle process of a component instance.

When a component instance is `created` , the created lifecycle is triggered. In `this case` , the `component data` , **this.data is defined as data in the Component constructor**. You cannot call setData yet. Generally, this lifecycle should only be used to add custom property fields to the component this.

After the component has been completely initialized and entered the page node tree, the `attached` lifecycle is triggered. In this case, `this.data` has been initialized to the current value of the component. This lifecycle is very useful, where most initialization work can be completed.

After the component leaves the page node tree, the `detached` lifecycle is triggered. When you exit a page, if the component is still in the page node tree, detached is triggered.

©2013-2025 Tencent Cloud International Pte. Ltd.

5.1 Defining Lifecycle Methods

Lifecycle methods can be directly defined in level-one parameters of the Component constructor.
component lifecycles can also be declared in the lifetimes field. We recommend this method as it has the highest priority.

Code example:

```
Component({
  lifetimes: {
    attached: function() {
      // Executed when the component instance enters the page node tree
    },
    detached: function() {
      // Executed when the component instance is removed from the page node tree
    },
  },
  // The older definition methods below can be used for compatibility with base lib
  attached: function() {
    // Executed when the component instance enters the page node tree
  },
  detached: function() {
    // Executed when the component instance is removed from the page node tree
  },
  // ...
})
```

You can also write lifecycle methods in behaviors, which will not be overwritten by lifecycles of the same name in other behaviors. Note that, if a component directly or indirectly references the same behavior multiple times, the lifecycle function in this behavior is only executed once during a single execution time.

All available lifecycles are listed below:

Lifecycle	Parameter	Description
created	None	Executed when the component instance is created
attached	None	Executed when the component instance enters the page node tree
ready	None	Executed when the component layout is completed in the view layer
moved	None	Executed when the component instance is moved to another node tree
detached	None	Executed when the component instance is removed from the page node tree
error	Object Error	Executed each time the component method throws an exception

5.2 Component Page Lifecycles

©2013-2025 Tencent Cloud International Pte. Ltd.

There are also some special lifecycles that are not strongly associated with components but must sometimes be known by components for internal processing. These lifecycles are called "component page lifecycles" and defined in the `pageLifetimes` definition field. The available component page lifecycles include:

Lifecycle	Parameter	Description
show	None	Executed when the component's page is displayed
hide	None	Executed when the component's page is hidden
resize	Object Size	Executed when the component's page is resized

Note：

The pageLifetime of the custom tabBar will not be triggered.

Code example:

```
Component({
  pageLifetimes: {
    show: function() {
      // Page displayed
    },
    hide: function() {
      // Page hidden
    },
    resize: function(size) {
      // Page resized
    }
  }
})
```

6. behaviors

The `behaviors` feature is used to share code between components, similar to "mixins" or "traits" in certain programing languages.

A `behavior` can include a set of properties, data, lifecycle functions, and methods. **When referenced by a component, the behavior's properties, data, and methods are merged into the component. Each lifecycle function is also called at the appropriate time**. A component can reference more than one behavior and a behavior can reference another behavior.

For specific definitions and usage of the parameters, see the Behavior.

6.1 Use in Components

When referenced by a component, behaviors are listed in the behaviors definition field.

Code example:

```
// my-behavior.js
module.exports = Behavior({
  behaviors: [],
  properties: {
    myBehaviorProperty: {
      type: String
    }
  },
  data: {
    myBehaviorData: {}
  },
  attached() {},
  methods: {
    myBehaviorMethod() {}
  }
})
```

When referencing components, just list them one by one in the behaviors definition section.

```
// my-component.js
var myBehavior = require('my-behavior')
Component({
  behaviors: [myBehavior],
  properties: {
    myProperty: {
      type: String
    }
  },
  data: {
    myData: 'my-component-data'
  },
  created: function () {
    console.log('[my-component] created')
  },
  attached: function () {
    console.log('[my-component] attached')
  },
  ready: function () {
    console.log('[my-component] ready')
  },
  methods: {
    myMethod: function () {
      console.log('[my-component] log by myMethod')
    },
```

```
    }
})
```

In the above example, my-behavior is added to the definition of the my-component component.

The structure of my-behavior is:

Property: myBehaviorProperty;

Data field: myBehaviorData;

method: myBehaviorMethod;

Lifecycle functions: attached, created, ready;

This will give my-component a final structure of:

Properties: myBehaviorProperty, myProperty;

Data fields: myBehaviorData, myData;

Methods: myBehaviorMethod, myMethod;

Lifecycle functions: attached, created, ready;

When the component triggers the lifecycle, the above example lifecycle functions are executed in the order:

1. [my-behavior] created;
2. [my-component] created;
3. [my-behavior] attached;
4. [my-component] attached;
5. [my-behavior] ready;
6. [my-component] ready;

For detailed rules, see Field Coverage and Combination Rules.

6.2 Field Coverage and Combination Rules

The component and the behaviour it references can contain fields with the same name, which are handled as follows:

If there are properties or methods with the same name:

If the component itself has the property or method, the property or method of the component overrides the property or method with the same name in the behavior;

If the component itself does not have this property or method, the property or method of the behaviour defined next in the behaviors field of the component overrides the property or method of the same name defined before it;

If there are nested references to behaviours in addition to 2, the rule is that the referencing behaviour overrides the property or method of the same name in the referenced behaviour.

If there are data fields with the same name:

If the data fields with the same name are both object types, object merging is performed;

In the rest of the cases, the data will be overwritten, and the overwriting rules are: referrer behaviour > referenced behaviour, backward behaviour > forward behaviour (higher priority overwrites lower priority, and the largest is the highest priority).

Lifecycle functions and observers do not override each other, but are called one by one at the corresponding trigger time:

For different lifecycle functions, the execution order of the component lifecycle functions is followed;
For the same kind of lifecycle function and the same field observers, the following rules are followed:
behaviour takes precedence over component execution;
The referenced behaviour takes precedence over the referrer behaviour;
The preceding behaviour takes precedence over the following behaviour;
If the same behaviour is referenced multiple times by a component, the lifecycle functions and observers defined by it will not be executed repeatedly.

6.3 Built-in Behaviors

Custom components can reference built-in behaviors to get some behaviors of built-in components.

Code example:

```
Component({
  behaviors: ['wx://form-field']
})
```

In the above example, `wx://form-field` represents a built-in behavior. It makes the custom component behave like a form control.
A built-in behavior generally adds properties to a component. Unless otherwise specified, components can overwrite these properties to change their type or add an observer.

6.3.1 wx://form-field

Make custom components behave like form controls. The form component recognises these custom components and returns the component's field name and its corresponding field value in the submit event.

Property	Type	Description
name	String	Field name in the form
value	Any	Field value in the form

Code sample:

```
// custom-form-field.js
Component({
  behaviors: ['wx://form-field'],
  data: {
    value: ''
  },
  methods: {
    onChange: function (e) {
      this.setData({
        value: e.detail.value,
```

```
      })
    }
  }
})
```

6.3.2 wx://component-export

Enables custom components to support the export definition field. This field can be used to specify the return value of the component when it is called by selectComponent.

Detailed usage and code examples can be found in the selectComponent.

7. Inter-component Relationship

7.1 Defining and Using Inter-component Relationship

Sometimes, you may need to implement components like this:

```
<custom-ul>
  <custom-li> item 1 </custom-li>
  <custom-li> item 2 </custom-li>
</custom-ul>
```

In this example, custom-ul and custom-li are custom components that are mutually related and communicate with each other in a complex manner. In this case, we can add the relations definition field when defining components.

Example:

```
// path/to/custom-ul.js
Component({
  relations: {
    './custom-li': {
      type: 'child', // The linked target node should be a child node
      linked: function(target) {
        // Executed each time custom-li is inserted. "target" is the instance objec
      },
      linkChanged: function(target) {
        // Executed each time custom-li is moved. "target" is the instance object o
      },
      unlinked: function(target) {
        // Executed each time custom-li is removed. "target" is the instance object
      }
    }
  },
  methods: {
    _getAllLi: function(){
      // You can get the nodes array via getRelationNodes, including all linked cus
```

```
      var nodes = this.getRelationNodes('path/to/custom-li')
    }
  },
  ready: function(){
    this._getAllLi()
  }
})
```

```
// path/to/custom-li.js
Component({
  relations: {
    './custom-ul': {
      type: 'parent', // The linked target node should be a parent node
      linked: function(target) {
        // Executed after each insertion to custom-ul. "target" is the instance obj
      },
      linkChanged: function(target) {
        // Executed after each move. "target" is the instance object of the custom-
      },
      unlinked: function(target) {
        // Executed after each removal. "target" is the instance object of the cust
      }
    }
  }
})
```

Note：

The relations definition must be added in the definitions of both components. Otherwise, it will be invalid.

7.2 Linking Components of the Same Class

Sometimes, you need to link components of the same class. For example:

```
<custom-form>
  <view>
    input
    <custom-input></custom-input>
  </view>
  <custom-submit> submit </custom-submit>
</custom-form>
```

The custom-form component wants to link the custom-input and custom-submit components. In this case, if the two components have the same behavior:

```
// path/to/custom-form-controls.js
module.exports = Behavior({
  // ...
```

```
})
```

```
// path/to/custom-input.js
var customFormControls = require('./custom-form-controls')
Component({
  behaviors: [customFormControls],
  relations: {
    './custom-form': {
      type: 'ancestor', // he linked target node should be an ancestor node
    }
  }
})
```

```
// path/to/custom-submit.js
var customFormControls = require('./custom-form-controls')
Component({
  behaviors: [customFormControls],
  relations: {
    './custom-form': {
      type: 'ancestor', // The linked target node should be an ancestor node
    }
  }
})
```

Then, in the relations definition, this behavior can be used to replace the component path as the linked target node:

```
// path/to/custom-form.js
var customFormControls = require('./custom-form-controls')
Component({
  relations: {
    'customFormControls': {
      type: 'descendant', // The linked target node should be a descendant node
      target: customFormControls
    }
  }
})
```

7.2 relations Definition Field

The `relations` definition field includes the target component path and its corresponding options. Available options are listed below.

Option	Type	Required	Description
type	String	Yes	The relative relationship of the target component, including parent, child, ancestor, and descendant.

linked	Function	No	A relationship lifecycle function, triggered when the relationship is created in the page node tree after the component's attached lifecycle.
linkChanged	Function	No	A relationship lifecycle function, triggered when the relationship is changed in the page node tree after the component's moved lifecycle.
unlinked	Function	No	A relationship lifecycle function, triggered when the relationship is removed from the page node tree after the component's detached lifecycle.
target	String	No	If this option is set, it indicates a behavior that should be possessed by the linked target node. All component nodes that have this behavior will be linked.

8. Data Listener

Data listener is used to monitor and respond to changes in any properties or data fields.

8.1 Using Data Listener

When certain data fields are set via setData, some operations must be executed.
For example, this.data.sum is always the sum of this.data.numberA and this.data.numberB. In this case, you can use a data listener to implement the following operation.

```
Component({
  attached: function() {
    this.setData({
      numberA: 1,
      numberB: 2,
    })
  },
  observers: {
    'numberA, numberB': function(numberA, numberB) {
      // Execute this function when numberA or numberB is set
      this.setData({
        sum: numberA + numberB
      })
    }
  }
})
```

8.2 Field Listening Syntax

Data listeners can monitor multiple changes to properties or internal data simultaneously. A listener can only be triggered one time by a setData operation.

Listeners can also monitor sub-data fields, as shown below.

```
Component({
  observers: {
    'some.subfield': function(subfield) {
      // Triggered when this.data.some.subfield is set via setData
      // (Also triggered when this.data.some is set via setData)
      subfield === this.data.some.subfield
    },
    'arr[12]': function(arr12) {
      // Triggered when this.data.arr[12] is set via setData
      // (Also triggered when this.data.arr is set via setData)
      arr12 === this.data.arr[12]
    },
  }
})
```

To monitor changes to all sub-data fields, you can use a wildcard `**` .

```
Component({
  observers: {
    'some.field.**': function(field) {
      // Triggered when setData is used to set this.data.some.field itself or any o
      // (Also triggered when this.data.some is set via setData)
      field === this.data.some.field
    },
  },
  attached: function() {
    // This will trigger the above observer
    this.setData({
      'some.field': { /* ... */ }
    })
    // This will also trigger the above observer
    this.setData({
      'some.field.xxx': { /* ... */ }
    })
    // This will still trigger the above observer
    this.setData({
      'some': { /* ... */ }
    })
  }
})
```

You can simply use the wildcard `**` to monitor all setData.

```
Component({
  observers: {
    '**': function() {
      // Triggered upon each setData operation
    },
  },
})
```

8.3 Bugs & Tips:

Data listeners monitor the data fields set via setData. Even if the values of these data fields do not change, the data listeners will still be triggered;
Be aware that, if you use setData to set the monitored data fields in the data listener function, this can create an infinite loop;
Compared to a property observer, a data listener is more powerful with better performance.

9. Data-only field

Pure data fields are data fields that are not used for interface rendering and can be used to improve page update performance.

9.1 Plain data fields in component data

In some cases, some fields in data (including those set by setData) are neither displayed on the interface nor passed to other components, but are only used internally by the current component.
In this case, you can specify such data fields as "pure data fields", which will only be recorded in this.data and will not be involved in any interface rendering process, which can help improve the page update performance.
The way to specify "pure data fields" is to specify pureDataPattern as a regular expression in the options definition section of the Component constructor, and fields whose names match this regular expression will become pure data fields.
Code example:

```
Component({
  options: {
    pureDataPattern: /^_/ // Specify all data fields starting with _ as pure data f
  },
  data: {
    a: true, // Plain data field
    _b: true, // Plain data field
  }, methods: { a: true, // Normal data field _b: true, // Plain data field
```

```
  methods: {
    myMethod() {
      this.data._b // Plain data fields can be retrieved in this.data
      this.setData({
        c: true, // Plain data field
        _d: true, // Plain data field
      })
    }
  }
})
```

Plain data fields in the above components will not be applied to WXML:

```
<view wx:if="{{a}}"> This line will be displayed </view>
<view wx:if="{{_b}}"> This line will not be shown </view>
```

9.2 Pure Data Fields in Component Properties

Properties can also be specified as pure data fields (regular expressions that follow the pureDataPattern).

A pure data field in a property can receive external incoming property values just like a normal property, but it cannot be used directly in the component's own WXML.

Code Example:

```
Component({
  options: {
    pureDataPattern: /^_/
  },
  properties: {
    a: Boolean,
    _b: {
      type: Boolean,
      observer() {
        // Don't do this! This observer will never be triggered
      }
    },
  }
})
```

Note：

The property observer is never triggered for data-only fields in properties! If you want to listen for property value changes, use a data listener instead.

9.3 Listening to Pure Data Fields with Data Listeners

data listener can be used to listen to plain data fields (as with normal data fields). In this way, the interface can be changed by listening and responding to changes in plain data fields.

The following example is a custom component that converts a JavaScript timestamp to a readable time.

Code example:

```
Component({
  options: {
    pureDataPattern: /^timestamp$/ // Specifying the timestamp attribute as a data-
  },
  properties: {
    timestamp: Number,
  },
  observers: {
    timestamp: function () {
      // When timestamp is set, display it as a readable time string.
      var timeString = new Date(this.data.timestamp).toLocaleString()
      this.setData({
        timeString: timeString
      })
    }
  }
})
```

```
<view>{{timeString}}</view>
```

10. Abstract Node

10.1 Using Abstract Nodes in Components

For a node in a custom component template, its corresponding custom component is not determined by the component itself, but the custom component caller. In this case, you can declare that this node is an "abstract node". For example, we can implement a selectable-group component by adding custom-radio or custom-checkbox to it. The wxml of this component is written as follows:

Code example:

```
<!-- selectable-group.wxml -->
<view wx:for="{{labels}}">
  <label>
    <selectable disabled="{{false}}"></selectable>
    {{item}}
  </label>
</view>
```

Where, "selectable" is not any component declared in the usingComponents field of the json file, but an abstract node. It must be declared in the componentGenerics field:

```
{
  "componentGenerics": {
    "selectable": true
  }
}
```

10.2 Using Components with Abstract Nodes

When using the selectable-group component, you must specify the specific component as "selectable":

```
<selectable-group generic:selectable="custom-radio" />
```

As such, when a selectable-group component instance is generated, a "custom-radio" component instance is generated for the "selectable" node. Likewise, if you do the following:

```
<selectable-group generic:selectable="custom-checkbox" />
```

A "custom-checkbox" component instance is generated for the "selectable" node.

Note：

The above-mentioned custom-radio and custom-checkbox must be included in the usingComponents definition field of the json file corresponding to this wxml.

```
{
  "usingComponents": {
    "custom-radio": "path/to/custom/radio",
    "custom-checkbox": "path/to/custom/checkbox"
  }
}
```

10.3 Default Components of Abstract Nodes

You can specify a default component for an abstract node. When no specific component is specified, an instance of the default component is created. The default component can be specified in the componentGenerics field:

```
{
  "componentGenerics": {
    "selectable": {
      "default": "path/to/default/component"
    }
  }
}
```

10.4 Tips

©2013-2025 Tencent Cloud International Pte. Ltd.

yyy in generic:xxx="yyy" referenced by the node's generic field must be a static value. Therefore, abstract nodes are not suitable for scenarios where the node name is determined dynamically.

11. Custom Component Extension

To better customize the functionality of custom components, you can use the custom component extension mechanism.

11.1 Effect of Extension

In order to better understand the effect of the extension, look at the example below:

```
// behavior.js
module.exports = Behavior({
  definitionFilter(defFields) {
    defFields.data.from = 'behavior'
  },
})

// component.js
Component({
  data: {
    from: 'component'
  },
  behaviors: [require('behavior.js')],
  ready() {
    console.log(this.data.from) // You can see that the output here is behavior ins
  }
})
```

As you can see from the example, extensions to custom components provide the ability to modify the data definition field in custom component.

11.2 Using Extensions

The Behavior() constructor provides a new definition field, definitionFilter to support the custom component extension. The definitionFilter function is injected with the following two parameters when it is called: the component/behavior definition object that uses this behavior, and the definitionFilter function list of the behavior used by this behavior. Here is an example:

```
// behavior3.js
module.exports = Behavior({
  definitionFilter(defFields, definitionFilterArr) {},
```

```
})

// behavior2.js
module.exports = Behavior({
behaviors: [require('behavior3.js')],
definitionFilter(defFields, definitionFilterArr) {
  // definitionFilterArr[0](defFields)
},
})

// behavior1.js
module.exports = Behavior({
behaviors: [require('behavior2.js')],
definitionFilter(defFields, definitionFilterArr) {},
})

// component.js
Component({
behaviors: [require('behavior1.js')],
})
```

The code above declares 1 custom component and 3 behaviors. Each behavior uses a definitionFilter definition field. Then, the following events occur in the order of the declaration:

1. When the behavior2 declaration is made, the definitionFilter function of behavior3 is called. The defFields parameter is the definition field of behavior2, and the definitionFilterArr parameter is an empty array because behavior3 does not use any other behavior.
2. When the behavior1 declaration is made, the definitionFilter function of behavior2 is called. The defFields parameter is the definition field of behavior1, the definitionFilterArr parameter is an array with a length of 1, and definitionFilterArr[0] is the definitionFilter function of behavior3 because behavior3 is used by behavior2. You can decide whether to call the definitionFilter function of behavior3 when the behavior1 declaration is made. If the call is needed, add the code definitionFilterArr[0](defFields) here and the definitionFilterArr parameter will be passed by the base library.
3. Similarly, when component is declared, the definitionFilter function of behavior1 is called.

To put it simply, the definitionFilter function can be interpreted as the fact that when A uses B, the A declaration calls the definitionFilter function of B and passes the definition object of A for B to filter. If B also uses C and D, it can decide whether to call the definitionFilter function of C and D to filter the defined object of A.

Code example:

The following is a simple implementation of the Calculated Properties feature of the custom component using extensions:

```
// behavior.js
module.exports = Behavior({
  lifetimes: {
```

```
    created() {
      this._originalSetData = this.setData // Original setData
      this.setData = this._setData // Packaged setData
    }
  },
  definitionFilter(defFields) {
    const computed = defFields.computed || {}
    const computedKeys = Object.keys(computed)
    const computedCache = {}

    // Calculate computed
    const calcComputed = (scope, insertToData) => {
      const needUpdate = {}
      const data = defFields.data = defFields.data || {}

      for (let key of computedKeys) {
        const value = computed[key].call(scope) // Calculate new value
        if (computedCache[key] !== value) needUpdate[key] = computedCache[key] = va
        if (insertToData) data[key] = needUpdate[key] // Insert the value directly
      }

      return needUpdate
    }

    // Override the setData method
    defFields.methods = defFields.methods || {}
    defFields.methods._setData = function (data, callback) {
      const originalSetData = this._originalSetData // Original setData
      originalSetData.call(this, data, callback) // Perform setData for data
      const needUpdate = calcComputed(this) // Calculate computed
      originalSetData.call(this, needUpdate) // Perform setData for computed
    }

    // Initialize computed
    calcComputed(defFields, true) // Calculate computed
  }
})
```

Used in components:

```
const beh = require('./behavior.js')
Component({
  behaviors: [beh],
  data: {
    a: 0,
  },
  computed: {
```

```
    b() {
      return this.data.a + 100
    },
  },
  methods: {
    onTap() {
      this.setData({
        a: ++this.data.a,
      })
    }
  }
})
```

```
<view>data: {{a}}</view>
<view>computed: {{b}}</view>
<button bindtap="onTap">click</button>
```

The implementation principle is simple: Perform a secondary encapsulation for an existing setData, calculate the value of each field in computed each time you perform setData, and then set it into the data to implement computed property.

12. Placeholder

When using features such as the Packet Asynchronisation feature, other custom components referenced by the custom component may be unavailable at the start of the rendering process. In this case, in order for the rendering process not to be blocked, the unavailable custom component needs a Component placeholder. The base library renders the unavailable component with the placeholder, and then replaces the placeholder with the component when it becomes available.
The placeholder for a custom component can be another custom component, or a built-in component.

12.1 configure

The componentPlaceholder field in the JSON configuration corresponding to the page or custom component is used to specify the placeholder component, for example:

```
{
  "usingComponents": {
    "comp-a": "../comp/compA",
    "comp-b": "../comp/compB",
    "comp-c": "../comp/compC"
  },
  "componentPlaceholder": {
    "comp-a": "view",
```

```
    "comp-b": "comp-c"
  }
}
```

This configuration indicates:
the placeholder component for component comp-a is the built-in component view;
The placeholder component for component comp-b is the custom component comp-c (the path to which is configured in usingComponents).
Assume that the template corresponding to this configuration is as follows:

```
<button ontap="onTap">display component</button>
<comp-a wx-if="{{ visible }}">
  <comp-b prop="{{ p }}">text in slot</comp-b>
</comp-a>
```

If visible is false when the mini program starts, only the button will be rendered; when the button is clicked, this.setData({ visible: true }) is executed, and if comp-a, comp-b are not available, the page will be rendered as:

```
<button>display component</button>
<view>
  <comp-c prop="{{ p }}">text in slot</comp-c>
</view>
```

When comp-a and comp-b are ready, the page is replaced:

```
<button>display component</button>
<comp-a>
  <comp-b prop="{{ p }}">text in slot</comp-b>
</comp-a>
```

12.2 Cautions

1. When a component is specified as a placeholder component (such as comp-c in the above example), it is not valid to specify a placeholder component for it. It is understood that if a component needs to be used as a placeholder component for other components, it must be available in the first place;
2. Currently custom components unavailable include:
2.1 In the case of using the subpackage asynchronisation feature, a component from another subpackage is referenced and the corresponding subpackage has not yet been downloaded;
2.2 the component has not yet been injected in the case of the use of the inject-when-injected feature;
3. If a component is not available and its placeholder component does not exist, an error will be reported and thrown when rendering;
4. if a component does not exist, but an available placeholder component is specified for it, the placeholder component can be rendered normally, but subsequent attempts to prepare a replacement will report an error and throw.

©2013-2025 Tencent Cloud International Pte. Ltd.

Attachment: Rendering flow with placeholder components involved

When the base library tries to render a component, it first recursively checks usingComponents to collect information about all the components it will use; in this process, if a component being used is not available, the base library will first check if it has a corresponding placeholder component. If not, the library interrupts the rendering and throws an error; if it does, it flags the unavailable component and replaces it with a placeholder component in the subsequent rendering process. The unavailable component will try to prepare (download a subpackage or inject code, etc.) at the end of the current rendering process; when the preparation process is complete, it will try to render the component again (in effect, performing the same process) and replace the previously rendered placeholder component.

©2013-2025 Tencent Cloud International Pte. Ltd.

Basic Competencies
Network

Last updated：2025-03-25 18:15:55

Note：
When using web-related APIs in Mini Programs/Mini Games, developers should note the following issues in advance.

Configuration of Server Domain Name

Each mini program needs to set a communication domain name in advance, and the mini program can only communicate with the specified domain name. This includes normal HTTPS requests (wx.request), uploading files (wx.uploadFile), downloading files (wx.downloadFile) and Websocket communication (wx.connectSocket), UDP communication (wx.createUDPSocket), TCP communication (wx.createTCPSocket).
The network interface only allows communication with non-local IPs and configured server domains on the same LAN.

Configuration process

Configure the server domain name in Mini Program Console > Settings > Development Settings > Server Domain Name. Configuration notes:
The domain name supports the https (wx.request, wx.uploadFile,wx.downloadFile) protocol, the wss (wx.connectSocket) protocol, and the tcp and udp protocols;
The domain name cannot use an IP address or localhost.
Ports can be configured, such as https://miniApp.com:8080, but when configured, requests can only be made to https://miniApp.com:8080. Requests to URLs such as https://miniApp.com, https://miniApp.com:9091, etc. will fail.
If no port is configured. For example, https://miniApp.com, then the request URL cannot contain a port, not even the default port 443. Requests to https://miniApp.com:443 will fail.

Network Request

Timeout

The default timeout and maximum timeout are both 60s;
The timeout can be configured in app.json;
It is also possible to specify a timeout when the interface is called, e.g. wx.request({ timeout: 5000 }) in ms. The timeout configuration for the interface call takes precedence over the configuration in app.json.

Use Limits

The referer header of a network request is optional. Its format is always https://servicewechat.com/{appid}/{version}/page-frame.html, in which {appid} is the Mini Program's appid and {version} is the version number of the Mini Program. A version number of 0 means it is a developer version, trial

version, or review version. A version number of devtools means it is the Weixin DevTools. Those with a version number other than the above ones are formal versions;
The maximum concurrency of wx.request, wx.uploadFile,wx.downloadFile is 10;
The maximum concurrency limit for wx.connectSocket is 5;
When the Mini Program is running in the background, if a network request does not end within 5s, an error message of fail interrupted will be called back. The network request API cannot be called until the Mini Program client is re-opened.

Return value encoding

The values returned by the server should be UTF-8 encoded. The Mini Program will try to convert non-UTF-8 encoded return values but may fail;
The Mini Program automatically filters the BOM header (only one BOM header is filtered).

Callback function

As long the value returned by the server is received, the success callback is triggered regardless of statusCode. Determine the return value based on the business logic.

FAQs

HTTPS certificate

Mini Programs must use HTTPS/WSS to initiate a network request. During the request, the system verifies the HTTPS certificate used by the server domain name. If the verification fails, the request is not successfully initiated. Due to system limitations, different platforms have different levels of stringency for certificates. To ensure the compatibility of Mini Programs, we recommend that developers configure the certificate according to the highest standard and use the relevant tools to check whether the existing certificate meets the requirements.
The requirements for certificates are as follows:
The HTTPS certificate must be valid;
Use a trusted certificate, i.e., the root certificate has been built into the system;
The domain name of the website where the SSL certificate is deployed must be the same as the domain name for which the certificate was issued;
The certificate must not have expired;
The certificate's trust chain must be complete (server configuration is required);
Self-signed certificates is not supported for iOS;
Certificates in iOS must meet Apple's App Transport Security (ATS) requirements;
TLS 1.2 or above is required. However, TLS 1.2 is not supported on some older Android models, so make sure HTTPS server supports TLS 1.2 or below;
Some CAs may not be trusted by the operating system. Take note of the announcements of the Mini Program and other systems when selecting certificates.
Note：

The validity of the certificate can be verified using the openssl s_client -connect example.com:443 command, or you can use other online tools;
In addition to the web request API, if there is any abnormality in other HTTPS requests in the mini program, please check it according to the above process. For example, https images can't be loaded, audio and video can't be played, and so on.

Skip domain name verification

In the Weixin DevTools, the Do not verify the request domain name, TLS version, and HTTPS certificate in development environment option can be enabled to skip verification of the server domain name. In this case, the server domain name will not be verified in the Weixin DevTools or when debugging mode is enabled on your phone.
After the server domain name is configured, we recommend you disable this option during development and test the domain name on each platform to ensure it is correct.
Note：
If you see the phenomenon of "can send request with debugging mode on, can't send request with debugging mode off" on your mobile phone, please make sure that the domain name verification has been skipped, and make sure that the server domain name and certificate configuration are correct.
If you encounter "internal server error", please try the following methods to check yourself ① Change the browser ② Refresh the web page ③ Check whether the file is placed in the correct server root directory. Checking method: Splicing the URL according to the following rules, and try to access whether it is successful or not: https://m.da9c.cn/Filename.

Storage

Last updated：2024-11-21 18:34:18

Each Weixin Mini Program has its own local cache. You can read, write, and clear the local cache via wx.setStorage/wx.setStorageSync, wx.getStorage/wx.getStorageSync, wx.clearStorage/wx.clearStorageSync, and wx.removeStorage/wx.removeStorageSync.

Isolation Policy

The storage limit of a Mini Program is 10 MB for a Weixin user. Storage is isolated based on users. On the same device, user A cannot read the data of user B. A Mini Program's user cannot read or write data of another Mini Program.

Erasion Rules

Local cache files are erased only when the code package is erased.

©2013-2025 Tencent Cloud International Pte. Ltd.

File System

Last updated：2025-05-08 11:10:01

The file system consists of a set of storage areas isolated based on Mini Programs and users and a set of appropriate management APIs. A globally unique file system manager is obtained via wx.getFileSystemManager(). The management operations for all file systems are called via FileSystemManager.

```
var fs = wx.getFileSystemManager()
```

Files are divided into two categories:
Code package file: A file added in the project directory;
Local file: A file generated locally by calling an API, or downloaded over the network and stored locally;
Local files are divided into three types:
Local temporary file: A file that is temporarily generated and will be recycled at any time. A maximum of 4 GB can be stored during runtime. After the runtime ends, if more than 2 GB has been used, the files will be cleaned up in descending order of most recent use time until the minimum of 2 GB is reached.
Local cache file: A file generated by a mini program after caching a local temporary file via an API. The directory and file name cannot be customized. Together with local user files, mini-programs (including mini-games) can store up to 200MB.
Local user file: A file generated by a mini program after caching a local temporary file via an API. The directory and file name can be customized. Together with local user files, mini-programs (including mini-games) can store up to 200MB.

1. Code Package File

Due to limited size, code package files are those needed for the first load. Large files and those that require dynamic replacement should not be added to the code package, but should be downloaded to the local device via the download API after the Mini Game is launched.

1.1 Access code package files

Code package files are accessed by writing the file path from the project root directory, relative path writing is not supported. For example: `/a/b/c` , `a/b/c` are legal, `. /a/b/c` , `. /a/b/c` are not legal.

1.2 Modify code package files

Files in the code package cannot be dynamically modified or deleted after running. Modifying the code package files requires a version re-release.

©2013-2025 Tencent Cloud International Pte. Ltd.

2. Local File

Local file refers to an individual file storage isolated based on users after the Mini Program is added to the mobile device by the user. A Weixin user cannot access the files of other users who have logged in on the same phone, and a user who have multiple appIds cannot access files under one appId using another appId.
The file path of a local file follows this format:

```
{{Protocol Name}}://File Path
```

Note：
The protocol name is "wxfile" in the Weixin app for iOS/Android and "http" in the Weixin DevTools. Developers can ignore this difference, and should not hard code a complete file path in the code.

2.1 Local temporary file

Local temporary file can only be generated by calling a specific API. However, you do not have the permission to write to the file directly. After a local temporary file is generated, it is valid only during the Mini Program's current lifecycle and is not available after the Mini Program restarts. Therefore, you cannot save the local temporary file path for later use. To use a local temporary file later, you can convert it to a local cache file or a local user file via the FileSystemManager.saveFile() or FileSystemManager.copyFile() API.
Code example:

```
wx.chooseImage({
  success: function (res) {
    var tempFilePaths = res.tempFilePaths // Each entry of tempFilePaths is a local
})
```

2.2 Local cache file

Local cache file can only be generated by calling a specific API and cannot be written directly. After a local cache file is generated, it is still available after the Mini Program restarts. Local cache files can only be obtained by saving the local temporary files via the FileSystemManager.saveFile() API.
Code example:

```
fs.saveFile({
  tempFilePath: '', // Pass a local temporary file path
  success(res) {
    console.log(res.savedFilePath) // res.savedFilePath is a local cache file path
  }
})
```

2.3 Local user file

Local user file is a new concept that has been added since version 1.7.0. We provide a user file directory for developers, who have completely free read/write access to this directory. The path to this directory is available via `wx.env.USER_DATA_PATH`.

Code example:

```
// Create a hello.txt file in the local user file directory and write "hello, world
const fs = wx.getFileSystemManager()
fs.writeFileSync(`${wx.env.USER_DATA_PATH}/hello.txt`,  'hello, world', 'utf8')
```

2.4 Read/Write Permission

API/Component	Read	Write
Code package file	Yes	No
Local temporary file	Yes	No
Local cache file	Yes	No
Local user file	Yes	Yes

2.5 Erasion Rules

Local temporary files are only guaranteed to persist during the Mini Program's current lifecycle. Once the Mini Program is closed, these files can be erased and may not be available at the next cold start;
Local cache files and local user files are erased only when the code package is erased.

Subpackage Loading

Last updated：2025-01-03 15:15:00

In certain scenarios, developers may find it necessary to partition the mini program into different sub-packages. During the construction phase, these are bundled into separate packages, which users can then load on an as-needed basis. When building a mini program with sub-packages, the construction process will yield one or more separate packages. Each mini program using sub-packages will invariably contain a main package. This main package houses the default launch page/TabBar page, along with any common resources/JS scripts required by all sub-packages. The division of sub-packages is determined based on the developer's configuration.
Upon the initiation of the mini program, the main package is downloaded by default and its internal pages are launched. When a user navigates to a page within a sub-package, the client will download the corresponding sub-package. Once the download is complete, the page is then displayed.
The current size limitations for mini program sub-packages are as follows:
The total size of all sub-packages within the mini program must not exceed 20MB.
The size of an individual sub-package or main package must not exceed 2MB.
Sub-packaging a mini program can optimize the initial download time at launch and enhance decoupled collaboration when multiple teams are working together.

1. Using Subpackage

1.1 Configuration

Assume that the directory structure of a Mini Program supporting subpackages is as follows:

```
├── app.js
├── app.json
├── app.wxss
├── packageA
│   └── pages
│       ├── cat
│       └── dog
├── packageB
│   └── pages
│       ├── apple
│       └── banana
├── pages
│   ├── index
│   └── logs
└── utils
```

The developer declares the subpackage structure of a project in the subpackages field via app.json:

> The field can also be written as subPackages.

```
{
  "pages":[
    "pages/index",
    "pages/logs"
  ],
  "subpackages": [
    {
      "root": "packageA",
      "pages": [
        "pages/cat",
        "pages/dog"
      ]
    }, {
      "root": "packageB",
      "name": "pack2",
      "pages": [
        "pages/apple",
        "pages/banana"
      ]
    }
  ]
}
```

In `subpackages`, each subpackage are configured with the following fields:

Field	Type	Description
root	String	Root directory of subpackage
name	String	Alias of subpackage, which can be used in Subpackage Preload
pages	StringArray	Subpackage page path, which is relative to the root directory of the subpackage
independent	Boolean	Indicates whether a subpackage is an Independent Subpackage

1.2 Packaging Rules

After subpackages is declared, files are packaged based on the paths configured in subpackages, and files in the directories outside the paths in subpackages are packaged into the app (main package);
The app (main package) can also have its own pages (i.e. the outermost pages field);

©2013-2025 Tencent Cloud International Pte. Ltd.

The root directory of subpackage cannot be a subdirectory configured in another subpackage;
The tabBar page must be in the app (main package).

1.3 Referencing Rules

packageA cannot require JS files in packageB, but can require JS files in app and packageA;
packageA cannot import templates in packageB, but can import templates in app and packageA;
packageA cannot use resources in packageB, but can use resources in app and packageA.

2. Independent Subpackage

An independent subpackage is a special type of subpackage in Mini Programs. It can run independently of the main package and other subpackages. When a user enters a Mini Program from a page of an independent subpackage, the main package will not be downloaded, unless the user enters a page of an ordinary subpackage or a main package.
Developers can add some functionally independent pages to an independent subpackage as needed. When a Mini Program is launched from a page of an ordinary subpackage, the main package should be downloaded first. However, an independent subpackage can run independently of the main package. Therefore, subpackages greatly improve the startup speed of Mini Programs.
A Mini Program can have multiple independent subpackages.

2.1 Configuration

Assume that the directory structure of a Mini Program is as follows:

```
├── app.js
├── app.json
├── app.wxss
├── moduleA
│   └── pages
│       ├── rabbit
│       └── squirrel
├── moduleB
│   └── pages
│       ├── pear
│       └── pineapple
├── pages
│   ├── index
│   └── logs
└── utils
```

The developer declares a subpackage as an independent subpackage by defining the independent field in the subpackage configuration item in the `subpackages` field of `app.json` .

```
{
  "pages": [
    "pages/index",
    "pages/logs"
  ],
  "subpackages": [
    {
      "root": "moduleA",
      "pages": [
        "pages/rabbit",
        "pages/squirrel"
      ]
    }, {
      "root": "moduleB",
      "pages": [
        "pages/pear",
        "pages/pineapple"
      ],
      "independent": true
    }
  ]
}
```

2.2 Limits

The independent subpackage is a type of subpackage. All limits on ordinary subpackages are applicable to independent subpackages. Plug-ins and custom components in an independent subpackage are processed in the same way as with those in an ordinary subpackage:
Also note the following when using independent subpackages:
The content of an independent subpackage must be independent of the main package and other subpackages, including JS files, template, wxss, custom components, and plug-ins. app.wxss in the main package does not take effect for independent subpackages. Therefore, the styles in app.wxss should be avoided on pages of independent subpackages;
App can be defined only in the main package. Defining App in independent subpackages will cause unexpected behaviors;
Plug-ins are not supported in independent subpackages.

2.3 Notes

2.3.1 About getApp()

Unlike ordinary sub-packages, when an independent subpackage is running, App is not necessarily registered. Therefore, an App object can not necessarily be obtained via getApp():
When a user starts a Mini Program from a page of an independent subpackage, neither main package nor App exists. In this case, undefined is obtained when getApp() is called. The main package is downloaded and App can be registered only when the user opens a page of an ordinary subpackage or a main package.
When a user is redirected to a page of an independent subpackage from a page of an ordinary subpackage or the main package, the main package already exists, and the actual App can be obtained by calling getApp().
Because of this restriction, developers can not share global variables between independent subpackages and other parts of a Mini Program by using the App object.
To satisfy this requirement in a standalone sub-package, getApp supports the [allowDefault] parameter, which returns a default implementation when the App is undefined. When the main package is loaded and the App is registered, the properties defined in the default implementation will be overridden and merged into the real App.
Sample code:
In the independent subpackage

```
const app = getApp({allowDefault: true}) // {}
app.data = 456
app.global = {}
```

In the app.js file

```
App({
  data: 123,
  other: 'hello'
})

console.log(getApp()) // {global: {}, data: 456, other: 'hello'}
```

2.3.2 About the lifecycle of App

If the Mini Program is launched from a page of an independent subpackage, onLaunch and first-time onShow of `App` in the main package are called when the user is redirected from the page of the independent subpackage to a page of the main package or an ordinary subpackage for the first time.
Because App cannot be defined in an independent subpackage, the lifecycle of a Mini Program can be listened via wx.onAppShow and wx.onAppHide. Other events on App can be listened via wx.onError and wx.onPageNotFound.

3. Subpackage Preload

Developers can configure the framework to automatically preload required subpackages when a user enters a Mini Program page, to speed up the subsequent startup of subSubcontracting Asynchronisationpackaged pages. For

Independent Subpackages, the main package can also be preloaded.
Subpackages can be preloaded only via configuration.

> Log prefixed with preloadSubpackages is available in vConsole to verify the preload status

3.1 Configuration

Subpackage preload is triggered when a user enters a specific page, and is controlled by adding `preloadRule` configuration items via `app.json` .

```
{
  "pages": ["pages/index"],
  "subpackages": [
    {
      "root": "important",
      "pages": ["index"],
    },
    {
      "root": "sub1",
      "pages": ["index"],
    },
    {
      "name": "hello",
      "root": "path/to",
      "pages": ["index"]
    },
    {
      "root": "sub3",
      "pages": ["index"]
    },
    {
      "root": "indep",
      "pages": ["index"],
      "independent": true
    }
  ],
  "preloadRule": {
    "pages/index": {
      "network": "all",
      "packages": ["important"]
    },
    "sub1/index": {
      "packages": ["hello", "sub3"]
    },
```

```
    "sub3/index": {
      "packages": ["path/to"]
    },
    "indep/index": {
      "packages": ["__APP__"]
    }
  }
}
```

In `preloadRule`, `key` is the page path, and `value` is the preload configuration for entering the page. Each configuration includes the following items:

Field	Type	Required	Default	Description
packages	StringArray	Yes	No	Indicates root or name of a subpackage preloaded after a user enters a page. __APP__ indicates the main package.
network	String	No	wifi	Indicates a specified network over which a subpackage is preloaded. Values: all: Any network wifi: Wi-Fi only

3.2 Limits

Pages of the same subpackage share the same maximum preload size of 2 MB, which is verified when the pages are packaged via the DevTools.
For example, Page A and Page B are in the same subpackage. If a subpackage with a total size of 0.5 MB is preloaded on Page A, a maximum of 1.5 MB subpackages can be preloaded on Page B.

4. Subcontracting Asynchronisation

In mini programs, different sub-packages correspond to different download units; therefore, except for non-independent sub-packages that can rely on the main package, sub-packages cannot use custom components or require each other. the "sub-package asynchronisation" feature will allow this limitation to be solved to a certain extent by allowing, through some configurations and new interfaces, some cross-sub-packages to wait for the download and then use the content asynchronously.

4.1 Cross-subcontract custom component references

When a sub-package uses a custom component from another sub-package, the other sub-package's component is in an unavailable state because the other sub-package has not yet downloaded or injected it. By setting a placeholder

component for the custom component of the other sub-package, we can render the placeholder component as a replacement first, and then replace it after the sub-package has finished downloading. Example:

```
// subPackageA/pages/index.json
{
  "usingComponents": {
    "button": "../../commonPackage/components/button",
    "list": "../../subPackageB/components/full-list",
    "simple-list": "../components/simple-list",
    "plugin-comp": "plugin://pluginInSubPackageB/comp"
  },
  "componentPlaceholder": {
    "button": "view",
    "list": "simple-list",
    "plugin-comp": "view"
  }
}
```

In this configuration, the button and list custom components are cross-package references, where the button is rendered using the built-in component view as a replacement, and the list is rendered using the current sub-package's custom component simple-list as a replacement; after these two sub-packages have been downloaded, the placeholder component is replaced with the corresponding cross-package component After both packages are downloaded, the placeholder component is replaced with the corresponding cross-package component.

4.2 Cross-Subpackage JS Code References

When code in a sub-package references code in another sub-package, we need to asynchronously get the result of the reference in order to prevent the download from blocking the code. For example:

```
// subPackageA/index.js
// Use a callback function style call
require('../subPackageB/utils.js', utils => {
  console.log(utils.whoami) // Wechat MiniProgram
}, ({mod, errMsg}) => {
  console.error(`path: ${mod}, ${errMsg}`)
})
// Or use a Promise-style call
require.async('../commonPackage/index.js').then(pkg => {
  pkg.getPackageName() // 'common'
}).catch(({mod, errMsg}) => {
  console.error(`path: ${mod}, ${errMsg}`)
})
```

See the Modularity - require section for more details.

Canvas

Last updated：2024-11-21 18:34:18

The Canvas provides a drawing interface on which you can draw anything you want.

1. Basic use

Step 1：Create a Canvas drawing context

```
<!-- 2d type canvas -->
<canvas id="myCanvas" type="2d" style="border: 1px solid; width: 300px; height: 150
```

First you need to add the Canvas to the WXML.

Specify `id="myCanvas"` to uniquely identify a canvas, which will be used to get the Canvas object later.

Specify `type` to define the type of the canvas, for this example, `type="2d"` is used.

Step 2: Use the Canvas drawing context to describe the drawing

```
wx.createSelectorQuery()
    .select('#myCanvas') // id in WXML
    .fields({ node: true, size: true })
    .exec((res) => {
        // Canvas Object
        const canvas = res[0].node
        // rendering context
        const ctx = canvas.getContext('2d')
    })
```

By selecting the canvas from the previous step via SelectorQuery, we can get the Canvas.

And through Canvas.getContext, we can get the rendering context RenderingContext.

Subsequent canvas operations and rendering operations need to be implemented through these two objects.

Step 3: Initialise the Canvas

```
wx.createSelectorQuery()
    .select('#myCanvas') // id in WXML
    .fields({ node: true, size: true })
    .exec((res) => {
        // Canvas Object
        const canvas = res[0].node
        // rendering context
```

```
        const ctx = canvas.getContext('2d')


        // The actual drawing width and height of the Canvas.
        const width = res[0].width
        const height = res[0].height


        // Initialise canvas size
        const dpr = wx.getWindowInfo().pixelRatio
        canvas.width = width * dpr
        canvas.height = height * dpr
        ctx.scale(dpr, dpr)
    })
```

The canvas width is divided into a rendered width and a logical width:

Rendered aspect is the actual width and height of the canvas canvas on the page, i.e. the size obtained by boundingClientRect requests to the nodes.

The logical width is the size of the logical width of the canvas during the rendering process, e.g. if you draw a rectangle with the same logical width, the rectangle will eventually fill the entire canvas. The logical width defaults to 300 * 150.

On different devices, there are cases where the physical pixels and logical pixels are not equal, so generally we need to use wx.getWindowInfo to get the pixel ratio of the device, multiply it by the rendered size of the canvas, and use it as the logical size of the canvas.

Step 4: Drawing

```
// Omit the initialisation steps above, the canvas object and ctx rendering context

// Empty the canvas
ctx.clearRect(0, 0, width, height)

// Drawing a red square
ctx.fillStyle = 'rgb(200, 0, 0)';
ctx.fillRect(10, 10, 50, 50);

// Drawing Blue Translucent Squares
ctx.fillStyle = 'rgba(0, 0, 200, 0.5)';
ctx.fillRect(30, 30, 50, 50);
```

The drawing api on the render context allows us to do arbitrary drawing on the canvas.

2. Advanced use

2.1 Drawing pictures

```
// Omit the initialisation steps above, the canvas object and ctx rendering context

// The image object
const image = canvas.createImage()
// Image load completion callback
image.onload = () => {
    // Draw the image to the canvas
    ctx.drawImage(image, 0, 0)
}
// Set the image src
image.src = 'https://open.weixin.qq.com/zh_CN/htmledition/res/assets/res-design-dow
```

With Canvas.createImage we can create an image object and load the image. Once the image is loaded and the onload callback is triggered, use ctx.drawImage to draw the image to the canvas.

2.2 Generate Image

```
// Omitting the initialisation steps above, the canvas object and the ctx rendering


// Drawing a red square
ctx.fillStyle = 'rgb(200, 0, 0)';
ctx.fillRect(10, 10, 50, 50);


// Drawing Blue Translucent Squares
ctx.fillStyle = 'rgba(0, 0, 200, 0.5)';
ctx.fillRect(30, 30, 50, 50);


// Generate Image
wx.canvasToTempFilePath({
    canvas,
    success: res => {
        // Path to the generated image temporary file
        const tempFilePath = res.tempFilePath
    },
})
```

The wx.canvasToTempFilePath interface allows you to generate a temporary image file from the contents of a canvas.

2.3 Frame animation

```
// Omit the initialisation steps above, the canvas object and the ctx rendering con

const startTime = Date.now()


// Frame rendering callbacks
const draw = () => {
  const time = Date.now()
  // Calculate the elapsed time
  const elapsed = time - startTime


  // Calculate the animation position
  const n = Math.floor(elapsed / 3000)
  const m = elapsed % 3000
  const dx = (n % 2 ? 0 : 1) + (n % 2 ? 1 : -1) * (m < 2500 ? easeOutBounce(m / 250
  const x = (width - 50) * dx


  // Rendering
  ctx.clearRect(0, 0, width, height)
  ctx.fillStyle = 'rgb(200, 0, 0)';
  ctx.fillRect(x, height / 2 - 25, 50, 50);


  // Register for next frame rendering
  canvas.requestAnimationFrame(draw)
}

draw()
```

With Canvas.requestAnimationFrame, you can register an animation frame callback and draw the animation frame by frame within the callback.

2.4 Custom Fonts

Custom fonts can be loaded for the Canvas via wx.loadFontFace.

Custom TabBar

Last updated：2024-11-28 10:34:10

Custom tabBar allows developers to flexibly set tabBar styles and achieve greater personalization.
In custom tabBar Mode:
In order to ensure compatibility with older versions and identify pages that are tab pages, the relevant tabBar configuration items must be completely declared. However, these fields cannot be used to render custom tabBars.
Developers must provide a custom component to render the tabBar, and all tabBar styles must be rendered via the custom component. To ensure a higher tabBar level, we recommend using cover-view and cover-image components which are fixed at the bottom of the page to render styles.
APIs related to tabBar styles, such as wx.setTabBarItem, will be deactivated.
The custom tabBar component instances vary with different tab pages. You can use the getTabBar API under a custom component to get the custom tabBar component instance of the current page.

Note：

To put a tab in the selected status, on the current page, use the getTabBar API to get the component instance and call setData to update the selected status. See the code sample at the bottom for details.

How to use

Configure information

Specify the custom field of the tabBar item in app.json and set the other configurations related to tabBar;
The usingComponents item must be declared in json of all tab pages. You can also enable this option globally in app.json.

Code example:

```
{
  "tabBar": {
    "custom": true,
    "color": "#000000",
    "selectedColor": "#000000",
    "backgroundColor": "#000000",
    "list": [{
      "pagePath": "page/component/index",
      "text": "Component"
    }, {
      "pagePath": "page/API/index",
      "text": "API"
    }]
  },
```

```
  "usingComponents": {}
}
```

Add tabBar code files

Add the entry files in the code root directory:

```
custom-tab-bar/index.js
custom-tab-bar/index.json
custom-tab-bar/index.wxml
custom-tab-bar/index.wxss
```

Write tabBar code

You can write the code using a custom component to render tabBar. In addition, the getTabBar API is added to the custom component to get the custom tabBar component instance of the current page.

```
Component({
  pageLifetimes: {
    show() {
      if (typeof this.getTabBar === 'function' &&
        this.getTabBar()) {
        this.getTabBar().setData({
          selected: 0
        })
      }
    }
  }
})
```

©2013-2025 Tencent Cloud International Pte. Ltd.

Multi-Threaded Worker

Last updated：2024-11-28 10:34:42

Some async tasks can be run in a worker. At the end of run, the results are returned to the main thread of the Mini Program. The worker runs in a separate global context and thread, and cannot directly call the methods of the main thread.
Data transfer between the worker and the main thread is accomplished using Worker.postMessage() to send data and Worker.onMessage() to receive data. The data to be transferred is not directly shared but copied.

Procedures

Configure Worker Information

In app.json, you can configure the directory where Worker code is placed. The code in the directory is packaged into a file:
Configuration example:

```
{
  "workers": "workers"
}
```

By configuring it in the above way, all the JS files under the workers directory will be packaged into one JS file and be part of the first package of the mini program.

Add Worker Code Files

Based on the configuration in Step 1, create the following two entry files:

```
workers/request/index.js
workers/request/utils.js
workers/response/index.js
```

After the files are added, the directory structure is shown as below:

```
├── app.js
├── app.json
├── project.config.json
└── workers
    ├── request
    │   ├── index.js
    │   └── utils.js
    └── response
        └── index.js
```

Write Worker Code

Write the worker response code in `workers/request/index.js` .

```
const utils = require('./utils')

// In the worker thread execution context, a worker object is globally exposed and
worker.onMessage(function (res) {
  console.log(res)
})
```

Initialize Worker in Main Thread

Initialize worker in app.js of the main thread.

```
const worker = wx.createWorker('workers/request/index.js') // The filename specifie
```

Main Thread Sends a Message to Worker

```
worker.postMessage({
  msg: 'hello worker'
})
```

Notes

The maximum concurrency of worker is one. Use Worker.terminate() to terminate the current worker before creating the next one;
The code in the worker can only require files in the specified worker path. It cannot reference other paths;
The worker entry files are specified when calling wx.createWorker(). Developers can dynamically specify the worker entry files;
wx APIs are not supported in workers;
Workers cannot send messages to each other.

DarkMode Adaptation Guide

Last updated：2024-11-21 18:34:18

1. Enable DarkMode

Configure darkmode as true in `app.json`, that means the current mini program has been adapted to DarkMode, all the basic components will display different default styles according to the system theme, and the navigation bar and tab bar will be automatically switched according to the following configuration.

2. Related Configuration

When `app.json` is configured with `darkmode` as `true`, some of the configuration items of mini program can be configured in the form of variables, defining the colour or icon under different themes in the variable configuration file, as follows:
1. Configure `themeLocation` in app.json to specify the path of variable configuration file theme.json, for example: add `theme.json` in the root directory, you need to configure `"themeLocation": "theme.json"`;
2. Define the relevant variables in theme.json;
3. Referencing variables in app.json starting with `@`;
Properties configured via variables are supported.
Globally configured window properties and properties under page configuration
navigationBarBackgroundColor;
navigationBarTextStyle;
backgroundColor;
backgroundTextStyle;
backgroundColourTop;
backgroundColorBottom.
Global configuration of window.tabBar properties
colour;
selectedColor;
backgroundColor;
borderStyle;
list;
iconPath;
selectedIconPath.

©2013-2025 Tencent Cloud International Pte. Ltd.

2.1 Variable configuration file theme.json

`theme.json` is used to define the variables related to the colour theme, you need to configure the path of `theme.json` in `themeLocation` first, otherwise you can't read the variable configuration.

The configuration file must contain the following properties:

Attributes	Type	Required	Descripción
light	object	yes	Definition of the variable in light mode
dark	object	yes	Definition of the variable in dark mode

Both `light` and `dark` can define variable names and values as key: value, for example:

```
{
  "light": {
    "navBgColor": "#f6f6f6",
    "navTxtStyle": "black"
  },
  "dark": {
    "navBgColor": "#191919",
    "navTxtStyle": "white"
  }
}
```

After completing the definition, it can be referenced in the global configuration or in the relevant properties of the page configuration starting with `@` , for example:

```
// Global Configuration
{
  "window": {
    "navigationBarBackgroundColor": "@navBgColor",
    "navigationBarTextStyle": "@navTxtStyle"
  }
}

// Page Configuration
{
  "navigationBarBackgroundColor": "@navBgColor",
  "navigationBarTextStyle": "@navTxtStyle"
}
```

After the configuration is completed, the mini program framework will automatically display the colour under the corresponding theme for the mini program according to the system theme.

2.2 Configuration Example

©2013-2025 Tencent Cloud International Pte. Ltd.

app.json (the example omits configuration items other than those related to the theme)

```
{
  "window": {
      "navigationBarBackgroundColor": "@navBgColor",
      "navigationBarTextStyle": "@navTxtStyle",
      "backgroundColor": "@bgColor",
      "backgroundTextStyle": "@bgTxtStyle",
      "backgroundColorTop": "@bgColorTop",
      "backgroundColorBottom": "@bgColorBottom"
  },
  "tabBar": {
      "color": "@tabFontColor",
      "selectedColor": "@tabSelectedColor",
      "backgroundColor": "@tabBgColor",
      "borderStyle": "@tabBorderStyle",
      "list": [{
          "iconPath": "@iconPath1",
          "selectedIconPath": "@selectedIconPath1"
      }, {
          "iconPath": "@iconPath2",
          "selectedIconPath": "@selectedIconPath2"
      }]
  }
}
```

theme.json

```
{
  "light": {
      "navBgColor": "#f6f6f6",
      "navTxtStyle": "black",
      "bgColor": "#ffffff",
      "bgTxtStyle": "light",
      "bgColorTop": "#eeeeee",
      "bgColorBottom": "#efefef",
      "tabFontColor": "#000000",
      "tabSelectedColor": "#3cc51f",
      "tabBgColor": "#ffffff",
      "tabBorderStyle": "black",
      "iconPath1": "image/icon1_light.png",
      "selectedIconPath1": "image/selected_icon1_light.png",
      "iconPath2": "image/icon2_light.png",
      "selectedIconPath2": "image/selected_icon2_light.png",
  },
  "dark": {
      "navBgColor": "#191919",
```

```
      "navTxtStyle": "white",
      "bgColor": "#1f1f1f",
      "bgTxtStyle": "dark",
      "bgColorTop": "#191919",
      "bgColorBottom": "#1f1f1f",
      "tabFontColor": "#ffffff",
      "tabSelectedColor": "#51a937",
      "tabBgColor": "#191919",
      "tabBorderStyle": "white",
      "iconPath1": "image/icon1_dark.png",
      "selectedIconPath1": "image/selected_icon1_dark.png",
      "iconPath2": "image/icon2_dark.png",
      "selectedIconPath2": "image/selected_icon2_dark.png",
  }
}
```

3. Get current system theme

If app.json declares `"darkmode": true`, `wx.getSystemInfo` or `wx.getSystemInfoSync` will return a `theme` attribute with a value of light or dark.
If app.json does not declare `"darkmode": true`, the theme property is not available (i.e. `theme` is `undefined`).

4. Listening for theme switching events

There are 2 ways to do this:
Pass onThemeChange callback method in App(), the callback will be triggered when the theme is switched;
Listen to theme change by wx.onThemeChange, wx.offThemeChange cancels the listening.

5. WXSS Adaptation

WXSS supports adapting different themes via prefers-color-scheme media query, which is the same as the web adapting method, for example:

```
/* General style begin */
.some-background {
  background: white;
}
.some-text {
```

```
  color: black;
}
/* General style end */


@media (prefers-color-scheme: dark) {
  /* Style start under DarkMode */
  .some-background {
      background: #1b1b1b;
  }
  .some-text {
      color: #ffffff;
  }
  /* Style end under DarkMode */
}
```

6. Tips

1. Note that the media query in WXSS is not affected by the darkmode switch in app.json. As long as the client supports DarkMode, the media query will take effect when the system is switched to DarkMode, no matter "darkmode": true is configured or not;
2. Theme switch event needs to be triggered when "darkmode": true is configured.

Open Capabilities
Banner Ad

Last updated：2025-04-09 18:35:49

To use mini program banner ads, you must first obtain ad authorization. For details, see Ad Authorization. Developers can create banner ads using the banner ad component. Once created, the component will automatically fetch and display ad data.

Ad size settings

Banner ads themselves cannot directly have their styles set through the style attribute. The default width is 100% (width: 100%), and the height is calculated proportionally. So developers can adjust the ad size by setting the width of the outer component. The width of the outer component must not be less than 300 px. If the width is less than 300 px, the banner ad width will be forcibly adjusted to 300 px.

WXML

WXSS

```
<view class="adContainer">
    <ad unit-id="xxxx"></ad>
</view>
```

```
/* The width of the outer component can be set to 100% or a specific value. */
.adContainer {  width: 100%;}
```

Ad event listening

After creation, the banner ad will be automatically fetched. Developers can use the onload and onerror events of the ad component to listen for successful or failed ad fetches.

WXML

JS

```
<view class="adContainer">
  <ad unit-id="xxxx" bindload="adLoad" binderror="adError" bindclose="adClose"></ad
</view>
```

```
Page({
  adLoad() {
    console.log('Banner ad loaded successfully')
  },
  adError(err) {
```

```
    console.log('Banner ad loading failed', err)
  }
})
```

Rewarded Video Ad

Last updated：2025-04-10 15:17:16

To use mini program rewarded ads, you must first obtain ad authorization. For details, see Ad Authorization.
The rewarded video ad component is composed of native image, text, and video controls from the client. It has the highest layer level and will cover regular components.
Developers can call wx.createRewardedVideoAd to create a rewarded video ad component. This method returns a singleton instance, which is only valid for the current page and cannot be used across pages.

Ad creation

The rewarded video ad component is hidden by default, allowing it to be pre-created for early initialization. Developers can create an ad instance in the onLoad event callback of the mini program page and repeatedly call the ad instance within the lifecycle of that page.

```
let rewardedVideoAd = null
Page({
  onLoad() {
    if(wx.createRewardedVideoAd){
      rewardedVideoAd = wx.createRewardedVideoAd({ adUnitId: 'xxxx' })
      rewardedVideoAd.onLoad(() => {
        console.log('onLoad event emit')
      })
      rewardedVideoAd.onError((err) => {
        console.log('onError event emit', err)
      })
      rewardedVideoAd.onClose((res) => {
        console.log('onClose event emit', res)
      })
    }
  }
})
```

To avoid misuse of ad resources, each user is limited to a certain number of rewarded video ads per day. It is recommended to check if the ad has been successfully fetched before displaying the ad button.

Show/Hide

The rewarded video ad component is hidden by default. After the user actively triggers the ad, the developer needs to call `RewardedVideoAd.show()` to display it.
Developers cannot control the hiding of the rewarded video ad component.

Fetch success and failure

```
rewardedVideoAd.onLoad(() => {
  console.log('Rewarded video ad loaded successfully')
})

rewardedVideoAd.show()
.then(() => console.log('Rewarded video ad displayed'))

rewardedVideoAd.onError(err => {  console.log(err)})
```

Listen for user rewards

Developers can listen to the onClose event to determine whether a user should receive a reward. The callback function for onClose will pass a res parameter, where res.isEnded describes the state when the ad ends. Developers need to check `res.isEnded` to determine if the video was watched to the end and then issue the reward to the user.

```
rewardedVideoAd.onClose(res => {
    // The user tapped the close button.
    if (res && res.isEnded) {
      // Normal playback ended, reward can be issued
    } else {
      // Playback was interrupted, no reward issued
    }
})
```

Note

Calling RewardedVideoAd.onLoad(), RewardedVideoAd.onError(), RewardedVideoAd.onClose(), and other methods multiple times to listen for ad events will result in multiple event callbacks. It is recommended to listen once after creating the ad, or to cancel the original listener before setting a new one.

©2013-2025 Tencent Cloud International Pte. Ltd.

Framework
Mini Program Configuration
Project Configuration

Last updated：2025-03-20 18:01:55

The `project.config.json` file in the project root directory allows you to configure the project.
The `project.config.json` file is mainly used in mini programs to configure the behaviour and characteristics of the mini program development tool. This file allows developers to adjust some of the default settings of the development tool to better suit the development process and environment.

configuration item

attribute	type	describe
miniprogramRoot	Path String	Specify the directory of the source code of the mini program (must be a relative path).
compileType	String	Compile type, default is miniprogram.
[xxx]appid	String	The appid of the project.
projectname	String	Project name
setting	Object	Project setting
libVersion	String	Base library version
condition	Object	Conditional Compilation Configuration
mockConfig	Object	IDE mockAPI Configuration

miniprogramRoot

The directory of the mini program source code (needs to be a relative path).

```
{
  "miniprogramRoot": "./src"
}
```

compileType

Compile type, defaults to miniprogram.

```
{
  "compileType": "miniprogram"
}
```

[xxx]appid

The appId corresponding to the mini program on the product console, xxx represents the product name.

```
{
  "[xxx]appid": "mpz5g4hpzd5p3k2l"
}
```

projectname

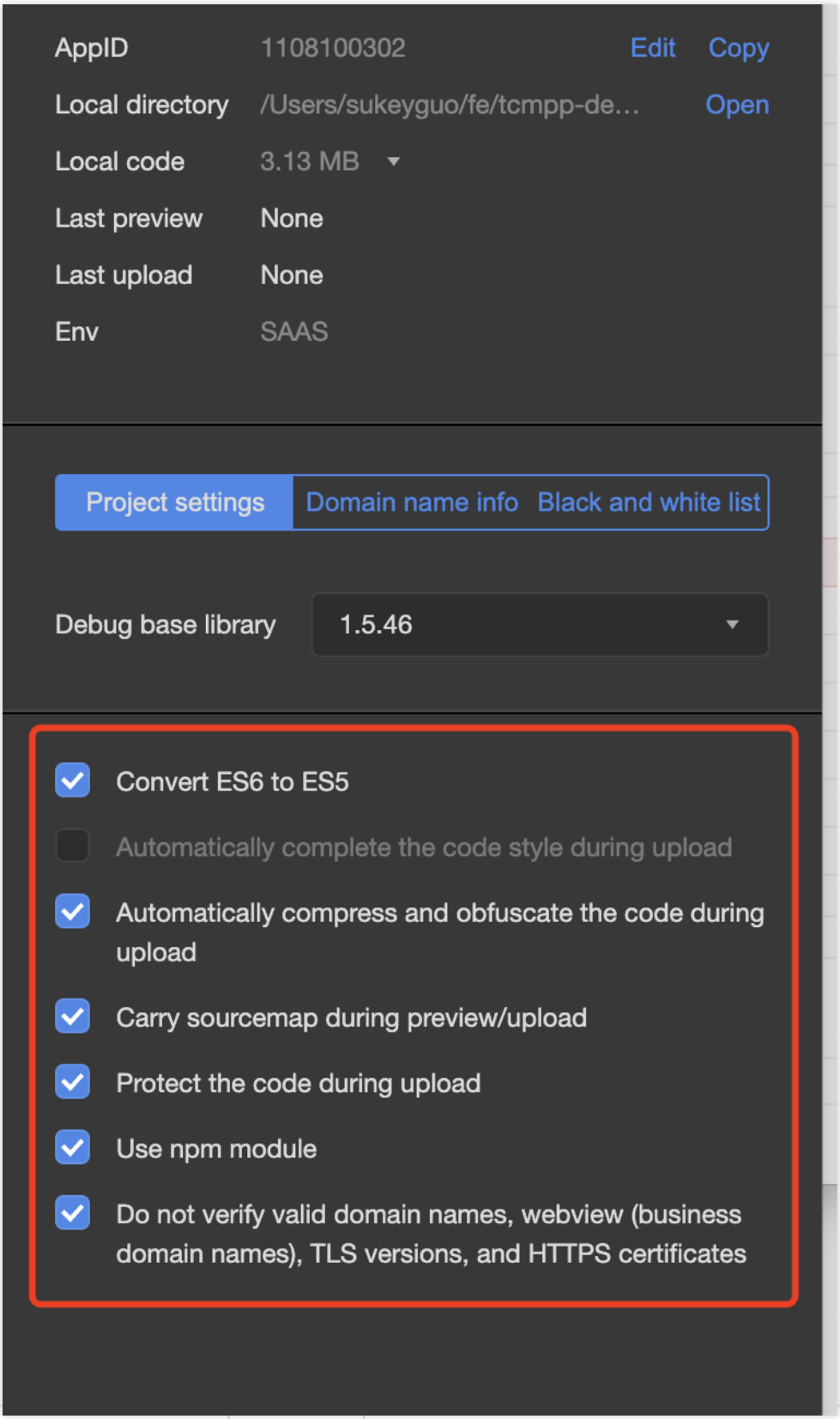
Project Title.

```
{
  "projectname": "main-demo"
}
```

setting

You can specify the following settings for the compilation settings of the project.

The following fields can be found in Developer Tools > Details > Local Settings.



attribute	type	describe
es6	Boolean	Whether to enable es6 to es5 conversion
postcss	Boolean	Whether to auto-complete the style when uploading code

minified	Boolean	If or not the script file will be automatically compressed when uploading code.
uglifyFileName	Boolean	Protect code when uploading
urlCheck	Boolean	Whether to check secure domain name and TLS version and HTTPS certificate

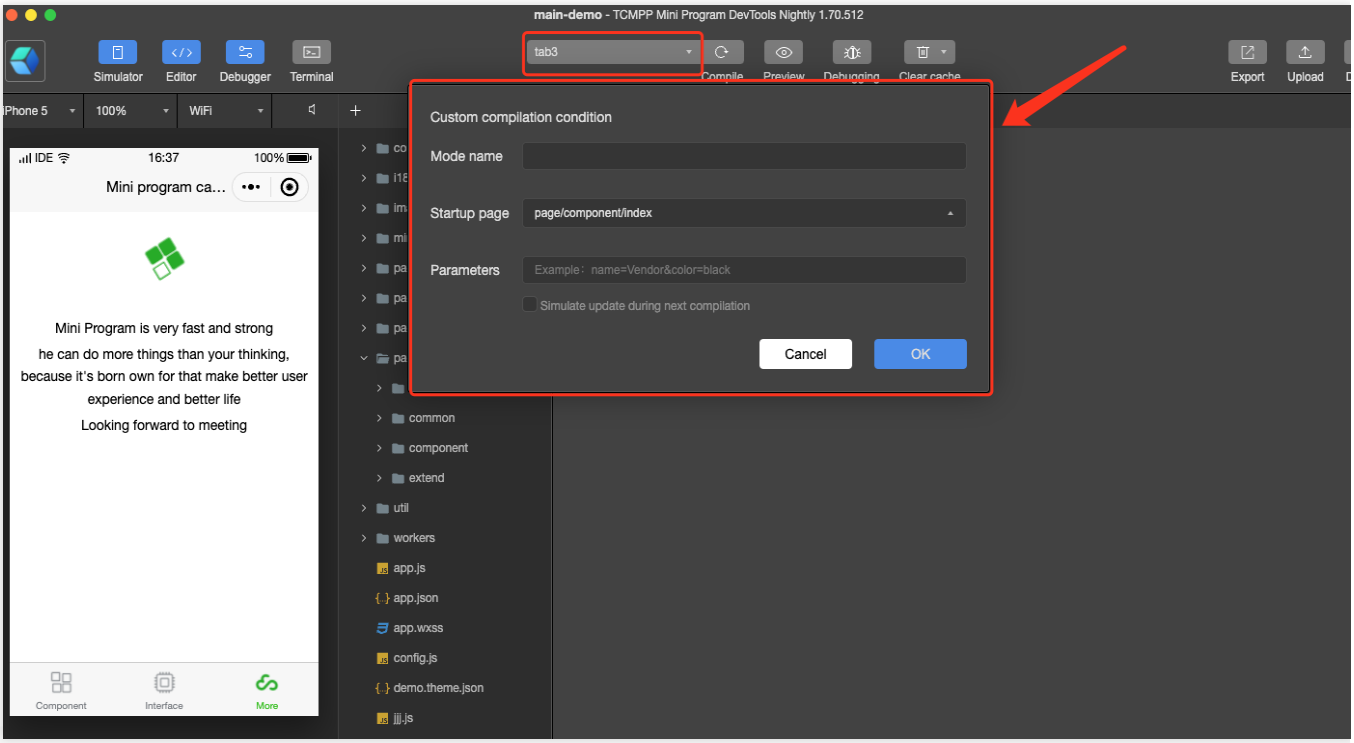
libVersion

The version number of the base library currently in use.

```
{
  "libVersion": "1.5.0"
}
```

condition

Corresponds to the conditional compilation information set on the IDE.

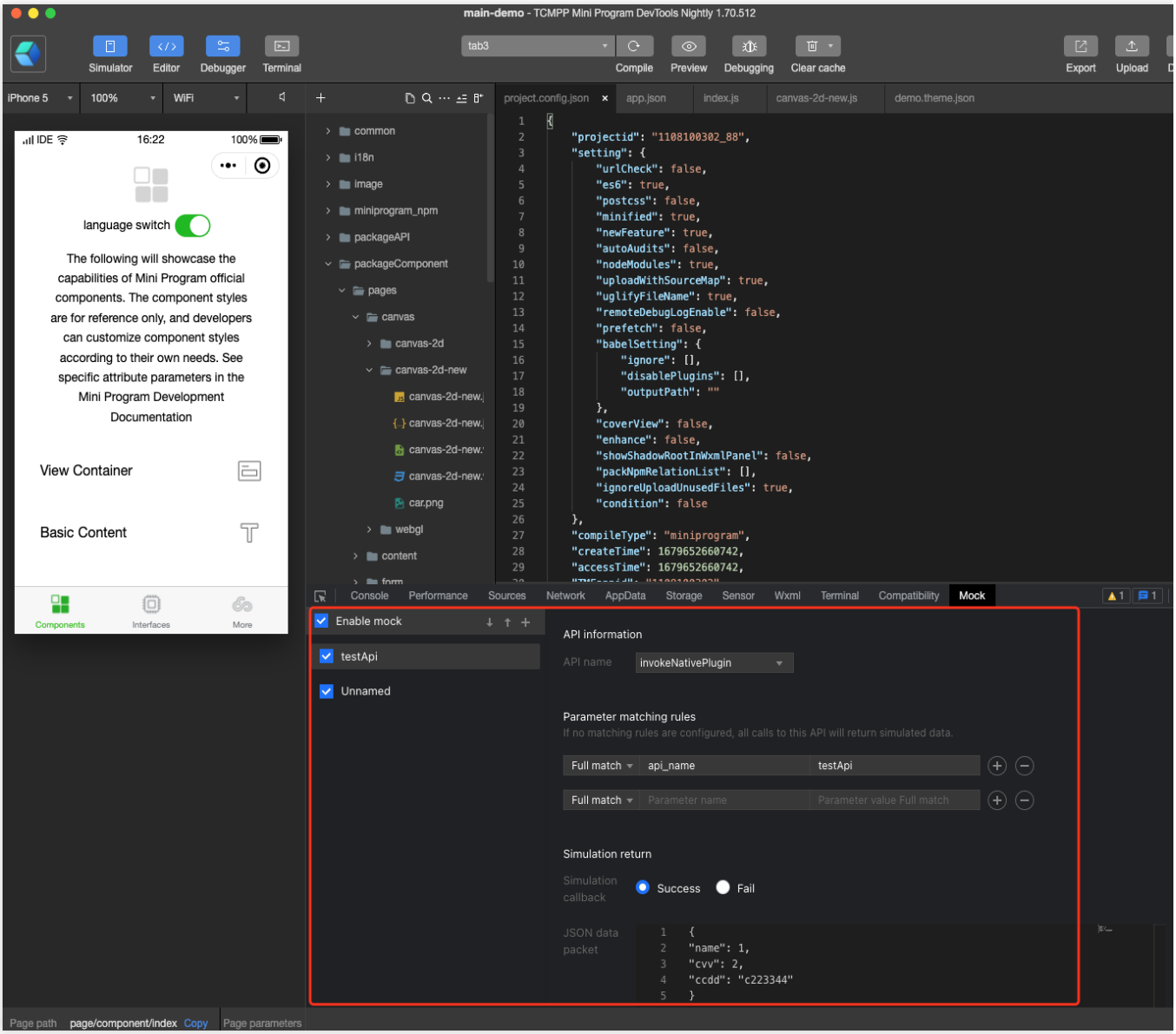



```
{
  "condition": {
    "miniprogram": {
      "current": 0,
      "list": [
        {
          "id": 0,
```

```
            "name": "tab3",
            "pathName": "page/extend/index",
            "query": "",
            "scene": null
          },
          {
            "id": -1,
            "name": "tab4",
            "pathName": "page/extend/index",
            "query": "name=1",
            "scene": null
          }
        ]
      }
    }
}
```

mockConfig

Configuration information that is automatically generated when the corresponding project mocks the API in the IDE debug panel.




```
{
  "globalOpen": true,
  "rules": [
    {
      "ruleName": "testApi",
      "ruleId": "48b83a70-84fd-400d-b53d-097518cfa515",
      "apiName": "invokeNativePlugin",
      "enable": true,
      "mockType": "API",
      "returnType": "succ",
      "resStr": "{\\n\\"name\\": 1,\\n\\"cvv\\": 2,\\n\\"ccdd\\": \\"c223344\\"\\n}
      "filterList": [
        {
          "propName": "api_name",
          "propRegString": "testApi",
          "matchType": "string"
```

```
        },
        {
          "propName": "",
          "propRegString": "",
          "matchType": "string"
        }
      ],
      "verifyFail": false
    }
  ]
}
```

Global Configuration

Last updated：2025-06-03 18:00:36

The `app.json` file in the root directory of the mini program is used for global configuration of the mini program. The file content is a JSON object with the following parameters:

Configuration item

attribute	type	required	describe
entryPagePath	string	no	Mini Program default start home page
pages	string[]	yes	Page Path List
window	Object	no	Global default window representation
tabBar	Object	no	BOTTOM tab Column performance
debug	boolean	no	Whether or not to open debug Mode, off by default
subpackages	Object[]	no	Subcontract structure configuration
workers	string	no	Worker Directory for code placement
navigateToMiniProgramAppIdList	string[]	no	The list of mini programs to jump to, see wx.navigateToMiniProgram for details.
preloadRule	Object	no	Subcontract Predownload Rules
usingComponents	Object	no	overall situation Custom Components configuration
permission	Object	no	Small Program Interface Permissions Related Settings
darkmode	boolean	no	Mini Program support DarkMode
themeLocation	string	no	Specify theme.json And dark mode true is required
disableSlideCloseGesture	boolean	no	Whether to disable the side swipe gesture to close the mini program. The default value is false.

©2013-2025 Tencent Cloud International Pte. Ltd.

apiVersion	string	no	Used for API version control (e.g., wx.login); if not configured, the latest version is called by default.

entryPagePath

Specifies the default startup path (homepage) of the mini program. If left blank, it will default to the first item in the pages list. Page path parameters are not supported.

```
{
  "entryPagePath": "pages/index/index"
}
```

pages

Specifies the pages that make up the mini-program, with each item corresponding to the path (including the filename) of a page. No need to include file extensions; the framework will automatically find and process the .json, .js, .wxml, and .wxss files.

If entryPagePath is not specified, the first item in the array represents the initial page (home page) of the mini-program.

To add/remove pages in the mini program, you need to modify the pages array.

For example, if the development directory is:

```
├── app.js
├── app.json
├── app.wxss
├── pages
│   ├── index
│   │   ├── index.wxml
│   │   ├── index.js
│   │   ├── index.json
│   │   └── index.wxss
│   └── logs
│       ├── logs.wxml
│       └── logs.js
└── utils
```

You need to app.json Middle write

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

window

Sets the status bar, navigation bar, title, and window background color of the mini-program.

attribute	type	default value	describe
navigationBarBackgroundColor	HexColor	#000000	Navigation bar background color, such as #000000
navigationBarTextStyle	string	white	Navigation bar header color, support only black / white
navigationBarTitleText	string	-	Navigation Title Text Content
navigationStyle	string	default	Navigation bar style supports only the following values **default** Default style, includes native navigation bar and capsule buttons. **custom** Customize the navigation bar, leaving only the capsule on in the upper right corner. It does not take effect on the page where the web-view is located. **hide** Hide the navigation bar and capsule buttons. It only hides the capsule buttons on the page where the web-view is located. **customV2** Customize the navigation bar, leaving only the capsule on in the upper right corner. **hideV2** Hide the navigation bar and capsule buttons. See note 2.
backgroundColor	HexColor	#ffffff	The background color of the window
backgroundTextStyle	string	dark	Pull down loading Style, only supports dark / light
backgroundColorTop	string	#ffffff	The background color of the top window, only iOS Support
backgroundColorBottom	string	#ffffff	The background color of the bottom window, only iOS Support
enablePullDownRefresh	boolean	false	Whether to enable global drop-down refresh. See for details Page.onPullDownRefresh
onReachBottomDistance	number	50	The distance from the bottom of the page when the event is triggered, in px. See for details Page.onReachBottom

customNavigateBack	Boolean	false	Whether or not you want to intercept the default return of the page (tabBar return or side-swipe or back key return) is used in conjunction with Page.onCustomBack.
pageOrientation	string	portrait	Screen rotation settings, support car / portrait / landscape See for details Response to display area changes

notes 1: HexColor (hexadecimal color value), such as "#ff00ff".

notes 2: The minimum version requirements for **customV2** and **hideV2** in the `navigationStyle` valid values are as follows:

IDE	Base Library	iOS SDK	Android SDK
2.2.611	2.2.6	2.2.6	2.2.6

tabBar

If the mini program is a multi-tab app (with a tab bar at the bottom or top of the client window for switching pages), you can use the tabBar configuration to define its appearance and the pages shown when tabs are switched.

attribute	type	required	default	describe
color	HexColor	yes	-	tab Default color for text on, only hexadecimal color is supported
selectedColor	HexColor	yes	-	tab Supports hexadecimal colors only
backgroundColor	HexColor	yes	-	tab Supports only hexadecimal colors
borderStyle	string	no	black	Tabs The color of the top border, Support only black / white
list	Array	yes	-	tab For a list of list Property explaination, minimum 2 Most 5 individual tab
position	string	no	bottom	tabBar Only supports the location of bottom / top
custom	boolean	no	false	custom Tab Bar, see you Custom tabBar

When a list accepts an array, **2 to 5 tabs can be configured**. The tabs are sorted in the order of the array, and each of them is an object with the following parameters:

attribute	type	required	introductions
pagePath	string	yes	Page path, must be in the pages First definition
text	string	yes	tab Up on text
iconPath	string	no	Picture path, icon Size limit is 40kb, recommended size 81px * 81px, does not support web pictures. when position for top Does not display when icon.
selectedIconPath	string	no	Selected picture path, icon Size limit is 40kb, recommended size 81px * 81px, does not support web pictures. when position for top Does not display when icon.

debug

You can enable the `debug` mode in the developer tool. In the console panel, debug information will be provided as `info`, including details on page registration, page routing, data updates, and event triggers. It helps developers quickly troubleshoot some common problems.

subpackages

When enabling Subpackage Loading,declare the project's subpackage structure.

Note：

SubPackages is also supported.

workers

Specifies the directory for placing Worker code when using Worker to handle multi-threaded tasks.

navigateToMiniProgramAppIdList

When the mini program needs to call the API to redirect to other mini programs, it is necessary to declare the list of up to 10 appId in the configuration file.

preloadRule

Declare the rules for Subcontract pre-download.

usingComponents

Custom components declared in `app.json` are considered global custom components and can be used directly in pages or custom components within the mini program without further declaration. It is recommended to only declare custom components that are used by almost all pages.

Note：

Global custom components will be considered dependencies for all pages and will be initialized when any page starts, affecting the startup performance and occupying the main package size. Custom components that are only used by specific pages or sub-packages should be declared in the page configuration whenever possible.

permission

Mini program API permission settings The field type is `Object` . Its structure is as follows:

attribute	type	required	default	describe
scope.userLocation	PermissionObject	no	-	Location-dependent permission statement

Permission Object structure

attribute	type	Required	Default	Introductions
desc	string	yes	-	Description of the Chinese usage of the interface displayed by the mini program when obtaining permissions. Maximum 80 characters
desc-en	string	no	-	Description of the English usage of the interface displayed by the mini program when obtaining permissions. Maximum 80 characters
desc-adesc-ar	string	no	-	A description of the Arabic usage of the interface displayed when the mini program gets permissions. Maximum 80 characters

Such as:

```
{
  "pages": ["pages/index/index"],
  "permission": {
    "scope.userLocation": {
      "desc": "Your location information will be used to display the effects of the
    }
  }
}
```

scope list

scope	API	Description
scope.userinfo	wx.getUserInfo	User information

scope.userLocation	wx.getLocation	Get location information
scope.userFuzzyLocation	wx.getFuzzyLocation	Get fuzzy location information
scope.record	live-pusher component or wx.startRecord	live-pusher component or wx.startRecord
scope.camera	camera component	camera component
scope.addPhoneCalendar	wx.addPhoneCalendar	Add Calendar
scope.writePhotosAlbum	wx.saveImageToPhotosAlbum	Save Image to Photos Album
scope.bluetooth	wx.openBluetoothAdapter	Bluetooth
scope.chooseMessageFile	wx.chooseMessageFile wx.chooseExternalFile	Select a file from the client session Select files from the file selector on your phone
scope.getPhoneNumber	button component: open-type="getPhoneNumber"	Get phone number
scope.getEmail	button component: open-type="getEmailAddress"	Get email address
scope.chooseImage	wx.chooseImage	Select images from the local photo album or take photos using the camera
scope.chooseVideo	wx.chooseVideo	Captures a video or selects a video from the phone's album
scope.chooseMedia	wx.chooseMedia	Captures or selects images or videos from the mobile phone's album

darkmode

By setting `"darkmode" to true`, you can enable dark mode in the mini program. All basic components will display different default styles according to the system theme, and the navigation bar and tab bar will also automatically switch according to the developer's configuration.

Please follow the DarkMode Adaptation Guide for further styling adjustments.

```
{
  "darkmode": true
}
```

themeLocation

The path to the custom theme.json. This configuration file is required when `"darkmode": true` is set.

disableSlideCloseGesture

Whether to disable the side swipe gesture to close the applet. The default value is false.

apiVersion

Used for API version control (e.g., wx.login); if not configured, the latest version is called by default.

```
{
  "apiVersion": "v1"
}
```

Configuration example

```
{
  "pages": ["pages/index/index", "pages/logs/index"],
  "window": {
    "navigationBarTitleText": "Demo"
  },
  "tabBar": {
    "list": [
      {
        "pagePath": "pages/index/index",
        "text": "index"
      },
      {
        "pagePath": "pages/logs/logs",
        "text": "logs"
      }
    ]
  },
  "debug": true
}
```

Page Configuration

Last updated：2025-04-29 18:40:51

Some of the configurations in app.json can also be configured for individual pages. You can configure the performance of the page in the corresponding .json file.
The configurations in the page will override the same configurations in app.json on the current page (style-related configurations are part of the window attribute in app.json, but you don't need to specify the window field here), and you can refer to the global configuration for the values and meanings of the configurations.
The content of the file is a JSON object with the following properties:

Configuration item

attribute	type	default	describe
navigationBarBackgroundColor	HexColor	#000000	Navigation bar background color, such as #000000
navigationBarTextStyle	string	white	Navigation bar header color, support only black / white
navigationBarTitleText	string		Navigation Title Text Content
navigationStyle	string	default	Navigation bar style supports only the following values **default** Default style, includes native navigation bar and capsule buttons. **custom** Customize the navigation bar, leaving only the capsule on in the upper right corner. It does not take effect on the page where the web-view is located. **hide** Hide the navigation bar and capsule buttons. It only hides the capsule buttons on the page where the web-view is located. **customV2** Customize the navigation bar, leaving only the capsule on in the upper right corner. **hideV2** Hide the navigation bar and capsule buttons. See note 1.
backgroundColor	HexColor	#ffffff	The background color of the window

backgroundTextStyle	string	dark	Pull down loading Style, only supports dark / light
backgroundColorTop	string	#ffffff	The background color of the top window, only iOS Support
backgroundColorBottom	string	#ffffff	The background color of the bottom window, only iOS Support
enablePullDownRefresh	boolean	false	Whether to enable the current page drop-down refresh. See for details Page.onPullDownRefresh
onReachBottomDistance	number	50	The distance, in px, from the bottom of the page when the pull-down event is triggered. See for details Page.onReachBottom
pageOrientation	string	portrait	Screen rotation settings, support car / portrait / landscape See for details Response to display area changes
disableScroll	boolean	false	Set to true The page cannot scroll up and down as a whole. Valid only in page configuration, not in app.json Set in
usingComponents	Object	no	page Custom Components configuration

notes 1: The minimum version requirements for **customV2** and **hideV2** in the `navigationStyle` valid values are as follows:

IDE	Base Library	iOS SDK	Android SDK
2.2.611	2.2.6	2.2.6	2.2.6

Note：

Not all configurations in app.json can be overridden or specified separately on the page, limited to the options included in this document.

Configuration example

```
{
```

```
  "navigationBarBackgroundColor": "#ffffff",
  "navigationBarTextStyle": "black",
  "navigationBarTitleText": "WeChat interface function demonstration,"
  "backgroundColor": "#eeeeee",
  "backgroundTextStyle": "light"
}
```

Framework Interface

App

Last updated：2025-03-20 18:01:55

App(Object object)

Sign up for the Mini Program. Take one. Object Parameter that specifies the lifecycle callbacks of Mini Programs, and so on.
App() Must be app.js Must be called and can be called only once. Or there will be unforeseen consequences.

parameter

Object object

attribute	type	default	required	Introductions
onLaunch	function	-	no	Lifecycle callback--Monitor Mini Program initialization.
onShow	function	-	no	Lifecycle callback--Monitor Mini Program start or swicth to the frontend.
onHide	function	-	no	Lifecycle callback--Monitor Mini Program swicth to the backend.
onError	function	-	no	Error listener.
onPageNotFound	function	-	no	There is no listener function on the page.
onUnhandledRejection	function	-	no	Untreated Promise Rejects event listener functions.
onThemeChange	function	-	no	Monitor subject changes
Other	any	-	no	Developers can add any function or data variable to Object Parameter, with the this Can access

For the definition of Mini Programs and the mechanism for running Mini Programs, see Operation mechanism.

©2013-2025 Tencent Cloud International Pte. Ltd.

onLaunch(Object object)

Mini Program initialization is completed when the trigger, the global trigger only once. Parameters can also be used wx.getLaunchOptionsSync Get.

parameter: With wx.getLaunchOptionsSync Agreement

onShow(Object object)

Mini Program start, or from the background into the foreground triggered when the display. You can also use wx.onAppShow Bind listening.

parameter: With wx.onAppShow Agreement

onHide()

Triggered when the Mini Program enters the background from the foreground. You can also use wx.onAppHide Bind listening.

onError(String error)

Script error or API Triggered on an error call. You can also use wx.onError Bind listening.

parameter: With wx.onError Agreement

onPageNotFound(Object object)

Triggered when the page the Mini Program is trying to open does not exist. You can also use wx.onPageNotFound Bind listening. Attention please refer to wx.onPageNotFound.

parameter: With wx.onPageNotFound Agreement

Sample code:

```
App({
  onPageNotFound(res) {
    wx.redirectTo({
      url: 'pages/...'
    }) // If it were Tabs Page, use the wx.switchTab
  }
})
```

onUnhandledRejection(Object object)

Mini Programs have unhandled Promise Triggered when rejected. You can also use wx.onUnhandledRejection Bind listening. Attention please refer to wx.onUnhandledRejection.

parameter: With wx.onUnhandledRejection Agreement

onThemeChange(Object object)

Triggered when the system switches topics. You can also use wx.onThemeChange Bind listening.

©2013-2025 Tencent Cloud International Pte. Ltd.

parameter: With wx.onThemeChange Agreement

The translations are provided by WeChat Translation and are for reference only. In case of any inconsistency and discrepancy between the Chinese version and the English version, the Chinese version shall prevail.Incorrect translation.

Sample code:

```
App({
  onLaunch (options) {
    // Do something initial when launch.
  },
  onShow (options) {
    // Do something when show.
  },
  onHide () {
    // Do something when hide.
  },
  onError (msg) {
    console.log(msg)
  },
  globalData: 'I am global data'
})
```

AppObject getApp(Object object)

Gets the Mini Programs globally unique App Instance.

parameter

Object object

attribute	type	default	required	Introductions
allowDefault	boolean	false	no	in App Returns the default implementation when undefined. When the app is invoked, properties defined in the default implementation are overridden and incorporated into the app. Generally usedIndependent subcontracting

Sample code

```
// other.js
```

```
var appInstance = getApp()
console.log(appInstance.globalData) // I am global data
```

Be careful

Don't define App() Or call the App Precall getApp() , using this You can ge it. app Instance;

adopt getApp() Do not privately call the lifecycle function after obtaining the instance.

Page

Last updated：2025-03-25 18:15:55

Page(Object object)

Register a page in the Mini Program. Take one. Object Type parameters that specify initial data for the page, lifecycle callbacks, event handlers, and so on.

parameter

Object object

attribute	type	default	required	Introductions
data	Object	-	-	Initial data for the page
onLoad	function	-	-	Lifecycle callbackMonitor page loading
onShow	function	-	-	Lifecycle callbackMonitor page display
onReady	function	-	-	Lifecycle callbackMonitor Page Initial Rendering Completed
onHide	function	-	-	Lifecycle callbackMonitor Page Hiding
onUnload	function	-	-	Lifecycle callbackListen Page Uninstall
onPullDownRefresh	function	-	-	Listen for user drop-down actions
onReachBottom	function	-	-	Handle function for bottoming event on page
onShareAppMessage	function	-	-	User click on the upper right to forward
onPageScroll	function	-	-	Handler for page scrolling trigger event
onTabItemTap	function	-	-	At present tab Page, click tab Time trigger
Other	any-	-	-	Other developers can add arbitrary functions or data to the Object parameter, which can be accessed in the page's function with this. This portion of the property is made a deep copy when the page instance is created.

sample code

```
//index.js
Page({
  data: {
    text: "This is page data."
  },
  onLoad: function(options) {
    // Do some initialize when page load.
  },
  onShow: function() {
    // Do something when page show.
  },
  onReady: function() {
    // Do something when page ready.
  },
  onHide: function() {
    // Do something when page hide.
  },
  onUnload: function() {
    // Do something when page close.
  },
  onPullDownRefresh: function() {
    // Do something when pull down.
  },
  onReachBottom: function() {
    // Do something when page reach bottom.
  },
  onShareAppMessage: function () {
    // return custom share data when user share.
  },
  onPageScroll: function() {
    // Do something when page scroll
  },
  onResize: function() {
    // Do something when page resize
  },
  onTabItemTap(item) {
    console.log(item.index)
    console.log(item.pagePath)
    console.log(item.text)
  },
  // Event handler.
  viewTap: function() {
    this.setData({
      text: 'Set some data for updating view.'
    }, function() {
      // this is setData callback
```

```
    })
  },
  customData: {
    hi: 'MINA'
  }
})
```

data

`data` is used in the first rendering of the pageInitial data。

When the page loads, `data` Will be `JSON` The string form is passed from the logical layer to the render layer, so `data` in must be converted toJSONType: string, number, boolean, object, array.

The render layer can be created by WXML Bind data.

Sample code:

```
<view>{{text}}</view>
<view>{{array[0].msg}}</view>
```

```
Page({
  data: {
    text: 'init data',
    array: [{msg: '1'}, {msg: '2'}]
  }
})
```

Lifecycle callback function

Triggering of the lifecycle and routing of the page Logic Layer.

onLoad(Object query)

Triggered when the page loads. A page will be called only once, and can be called in the onLoad Gets the parameters in the path to open the current page.

Parameters:

Name	type	Introductions
query	Object	Open parameters in the current page path

onShow()

Page display/Triggered when entering the front desk.

onReady()

Triggered when the initial rendering of the page is complete. A page is called only once, meaning the page is ready to interact with the view layer.
Note: to set the interface content API Such as wx.setNavigationBarTitle, in theonReadyAfter that. See for details Life cycle.

onHide()

Page concealment/Triggered when you switch to the backend. Such as wx.navigateTo Or bottom tab Switch to other pages, Mini Programs into the background, etc.

onUnload()

Triggered when the page is unloaded. Such as wx.redirectTo or wx.navigateBack to another page.

Page event handler

onPullDownRefresh()

Listen for user drop-down refresh events;
Need to app.json of window option, or Page Configuration enablePullDownRefresh;
Can be passed wx.startPullDownRefresh triggers a drop-down refresh, and then triggers a pull-down refresh animation. The effect is consistent with the user's manual pulldown refresh;
After processing the data refresh , wx.stopPullDownRefresh can stop the drop-down refresh of the current page.

onReachBottom()

Listen for user bottom-up events.
Can be app.json of window option, or Page Configuration onReachBottomDistance;
This event will be triggered only once during slippage within the trigger distance.

onPageScroll(Object object)

Listens for user sliding page events.
parameter Object object:

attribute	type	Introductions
scrollTop	Number	Distance a page has been scrolled vertically (in px)

Note:

Please only be present when needed. page Do not define an empty method. To reduce the impact of unnecessary event dispatch on rendering layer - logic layer communication.
Please avoid onPageScroll Too frequent execution in setData Equal cause Logic Layer and Render Layer Communication operations. In particular, each transmission of large amounts of data will affect the communication time.

©2013-2025 Tencent Cloud International Pte. Ltd.

onShareAppMessage(Object object)

Listen to the user click on the in-page forward on button assembly open-type="share"Or the upper-right menu "Forward" on behavior, and customize the forward content.

Note:

The Forward on is displayed on the upper-right menu only if this event handler is defined.

parameter Object object:

parameter	type	Introductions
from	String	Forward the source of the event. button:: In-page forward on menu:: Retweet menu on top right
target	Object	if from The value is buttonThen target That triggered this retweet. button, otherwise undefined
WebViewUrl	String	The page contains web-view, returns the current web-view
shareTarget	string	Where users click to share

This event handler function needs to return an Object, which is used to customise the forwarding content, the return content is as follows:

fields	type	description	default
title	string	Forwarding Title	Current mini program name
path	string	Forwarding Path	Current page path, must be a full path starting with /.
imageUrl	string	Custom image url, can be a local file url, a package file url or a network image url. Support PNG and JPG, display image aspect ratio is 5:4	Use default screenshot
entryDataHash	string	Listen to the user clicking the in-page forward button, only with this parameter, fast sharing will be supported	-
shareType	string	Specify the type of share	"miniapp"

Description of the shareTarget parameter:

Value	Description
0	Share to QQ friends
1	Share to QQ space
2	Open from current chat window, quickly share to current chat window
3	Share to WeChat friends
4	Share to WeChat Friends
5	Share panel to recent contacts
6	Share to Quick Share Friends List

Description of the shareType parameter:

Used to specify the type of sharing, currently there are two types of sharing, one is to share a mini program and the other is to share an image.

Value	Description
"miniapp"	Shared as a mini program, the title, path, imageUrl, entryDataHash parameters take effect
"picture"	Shared as a picture, imageUrl, entryDataHash parameters will take effect

Sample code:

```
Page({
  onShareAppMessage(res) {
    if (res.from === 'button') {
      // From the in-page forwarding button
      console.log(res.target)
    }
    return {
      title: 'Custom Forwarding Title',
      path: '/page/user?id=123'
    }
  }
})
```

onCustomBack()

Listen to the event of user back to previous page.

You need to enable customNavigateBack in the window option of `app.json` or in the Page Configuration;

This event will be triggered if you click tabBar to go back, or if you swipe or back key to go back.

onTabItemTap(Object object)

to hit tab Time trigger.

Object Parameter explaination:

parameter	type	Introductions
index	String	Number of tabItem clicked, starting with 0
pagePath	String	Page path of tabItem clicked
text	String	Click tabItem on text

Sample code:

```
Page({
  onTabItemTap(item) {
    console.log(item.index)
    console.log(item.pagePath)
    console.log(item.text)
  }
})
```

Component event handler

Page Component event handlers can also be defined in. In the components of the render layer Event binding is executed when an event is triggered Page The event handler defined in the.

Sample code:

```
<view bindtap="viewTap"> click me </view>
```

```
Page({
  viewTap: function() {
    console.log('view tap')
  }
})
```

Page.route

Path to the current page of typeString.

```
Page({
  viewTap: function() {
    console.log('view tap')
  }
})
```

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

setData Function is used to send data from the logical layer to the view layer (asynchronously), changing the corresponding this.data Is synchronized.

Parameter specification

field	type	Required	describe
data	Object	yes	The data to be changed this time
callback	Function	no	Interface update caused by setData after rendering

Object with key: value In the form of this.data to hit the target key The corresponding value is changed to value。among key Can be given in the form of a data path, supporting changing an item in an array or a property of an object, such as array[2].message， a.b.c.dAnd does not require this.data Predefined in.

Note：

Direct modification this.data Without calling this.setData Can not change the state of the page, but also cause data inconsistencies;

Supports settings only JSON Data;

A single set of data can not exceed 1024kB, try to avoid a set of too much data;

Please don't data Any of the value Set to undefined Otherwise this item will not be set and may leave some potential problems.

Sample code:

```
<!--index.wxml-->
<view>{{text}}</view>
<button bindtap="changeText"> Change normal data </button>
<view>{{num}}</view>
<button bindtap="changeNum"> Change normal num </button>
<view>{{array[0].text}}</view>
<button bindtap="changeItemInArray"> Change Array data </button>
<view>{{object.text}}</view>
<button bindtap="changeItemInObject"> Change Object data </button>
<view>{{newField.text}}</view>
<button bindtap="addNewField"> Add new data </button>
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
// index.js
Page({
  data: {
    text: 'init data',
    num: 0,
    array: [{text: 'init data'}],
    object: {
      text: 'init data'
    }
  },
  changeText: function() {
    // this.data.text = 'changed data' // Don't change this.data directly.
    // You should use setData
    this.setData({
      text: 'changed data'
    })
  },
  changeNum: function() {
    // Alternatively, you can modify this.data and then immediately set the modifie
    this.data.num = 1
    this.setData({
      num: this.data.num
    })
  },
  changeItemInArray: function() {
    // For object or array fields, you can modify a subfield directly under it, whi
    this.setData({
      'array[0].text':'changed data'
    })
  },
  changeItemInObject: function(){
    this.setData({
      'object.text': 'changed data'
    });
  },
  addNewField: function() {
    this.setData({
      'newField.text': 'new data'
    })
  }
})
```

PageObject[] getCurrentPages()

Get the current pages stack. The first element in the array is the first page and the last element is the current page.

Cautions

Don't try to modify the page stack, it will lead to routing and page state errors;

Don't call `getCurrentPages()` when `App.onLaunch` is running, the page has not been generated yet.

Custom Components

Last updated：2024-11-21 18:34:18

Component

Create a custom component that accepts a Object Parameter of type.

parameter

Object object

definition segment	type	Is it necessary to fill out	describe
properties	Object Map	no	External properties of a component are a map of property names to property settings
data	Object	no	Component's internal data, and properties Template rendering for components together
observers	Object	no	Component data field listener for listening properties and data See the changes in Data listener
methods	Object	no	Component, including event response functions and arbitrary custom methods. For the use of event response function, see Inter-component communication and events
behaviors	String Array	no	Inter-component code reuse mechanisms similar to mixins and traits, see behaviors
created	Function	no	Component lifecycle function - executes when the component instance has just been created, noting that it cannot be called at this time setData
attached	Function	no	Component life cycle function - executed when component instance enters page node tree
ready	Function	no	Component Life Cycle Function - Executed after component layout is complete
detached	Function	no	Component Lifecycle Function - Executes when component instance is removed from page node tree
relations	Object	no	Relationship definitions between components, see Inter-

			component relations
externalClasses	String Array	no	External style classes accepted by the component, see External style class
options	Object Map	no	Some options (specific option settings are covered when the features are described in the documentation, not listed here)
lifetimes	Object	no	Component lifecycle declaration object, see Component life cycle
pageLifetimes	Object	no	The lifecycle declaration object for the page on which the component resides, see Component life cycle

Generated component instances can be found in component methods, lifecycle functions, and properties `observer` Through this Access. Components contain common properties and methods.

Attribute name	type	describe
is	String	Component file path
id	String	Node id
dataset	String	Node dataset
data	Object	Component data,Including internal data and attribute values
properties	Object	Component data,Including internal data and attribute values(data Consistent)

Method name	parameter	describe
setData	Object newData	Set up data and perform view-layer rendering
triggerEvent	String name, Object detail, Object options	Trigger events, see Inter-component communication and events
createSelectorQuery	-	Create a SelectorQuery Object, the selector selects from within this component instance
createIntersectionObserver	-	Create a IntersectionObserver Object, the selector selects from within this component instance
SelecComponent	String selector	Select the component instance node using the selector and return the first component instance object that matches wx://component-export Impact

selectAllComponents	String selector	Select the component instance node using the selector and return an array of matched component instance objects wx://component-export Impact
selectOwnerComponent	String selector	Selects the component instance of the current component node (that is, the reference to the component) and returns its component instance object wx://component-export Impact
getRelationNodes	String relationKey	Gets all the associated nodes corresponding to this relationship, see Inter-component relations
hasBehavior	Object `behavior`	Checks if the component has a behaviour (the check recursively checks all behaviours that have been introduced directly or indirectly).

Code example:

```
Component({

  behaviors: [],

  // Property definition (see below for details)
  properties: {
    myProperty: { // Attribute name
      type: String,
      value: ''
    },
    myProperty2: String // Simplified definition
  },

  data:  {}, // Private data, available for template rendering

  lifetimes: {
    // Life cycle function, can be a function, or a method name defined in the meth
    attached: function () { },
    moved: function () { },
    detached: function () { },
  },

  // Life cycle function, can be a function, or a method name defined in the method
  attached: function () { }, // The attached declaration here is overwritten by the
  ready: function() { },

  pageLifetimes: {
    // Life Cycle Function for the Page on which the component is located
```

```
    show: function () { },
    hide: function () { },
    resize: function () { },
  },

  methods: {
    onMyButtonTap: function(){
      this.setData({
        // The method of updating properties and data is similar to that of updatin
      })
    },
    // The internal method recommends starting with an underscore
    _myPrivateMethod: function(){
      // Here will be data.A[0].B Set to 'myPrivateData'
      this.setData({
        'A[0].B': 'myPrivateData'
      })
    },
    _propertyChange: function(newVal, oldVal) {

    }
  }

})
```

properties definition

Definition segment	type	Is it necessary to fill out	describe
type	-	yes	Type of attribute
value	-	no	The initial value of the
observer	Function	no	Callback function when attribute value changes

Property values can be changed using the observer To monitor. Currently, this field is not recommended in the new version of the base library; instead, use the Component Constructor observers Field, which is more powerful and better performance.

Note：

The type field in the definition field is mandatory.

Code example:

```
Component({
  properties: {
    min: {
```

```
      type: Number,
      value: 0
    },
    min: {
      type: Number,
      value: 0,
      observer: function(newVal, oldVal) {
        // When property values change
      }
    },
    lastLeaf: {
      // This property can be Number 、 String 、 Boolean One of three types.
      type: Number,
      optionalTypes: [String, Object],
      value: 0
    }
  }
})
```

Property can be of type String Number Boolean Object Array One, can also be null Represents an unrestricted type. In most cases, it is best to specify an exact type for a property. So, in... WXML When an attribute value is specified literarily in, the value can obtain an exact type, such as:

```
<custom-comp min="1" max="5" />
```

At this point, because the corresponding property of the custom component is specified as Number Type, min and Max Is going to be assigned to 1 and 5 Instead of "1" and "5" , namely:

```
this.data.min === 1 //
truethis.data.Max === 5 // true
```

Tips:

Use this.data Can get internal data and attribute valuesBut modifying it directly does not apply changes to the interface, and you should use the setData Modify;

Lifecycle functions cannot be passed through component methods this Access to;

Property names should be avoided with data Beginning, that is, do not name dataXyz This form, because in WXML In, data-xyz="" Will be used as nodes dataset Instead of component properties;

In the definition and use of a component, the component's attribute name and data Fields cannot conflict with each other (although they are in different definition segments).

Behavior

Register a behaviorAccept one. Object Parameter of type.

parameter

Object object

Definition segment	type	Is it necessary to fill out	describe
properties	Object Map	No	Properties of the same component
data	Object	No	Properties of the same component
methods	Object	No	Same method as custom components
behaviors	String Array	No	For a mechanism for reusing code between components similar to mixins and traits, see behaviors.
created	Function	No	Component Lifecycle Functions - executed when the component instance has just been created, note that setData cannot be called at this time.
attached	Function	No	Component Lifecycle Functions - executed when the component instance enters the page node tree
ready	Function	No	Component Lifecycle Functions - executed after the component layout is complete
detached	Function	No	Component Lifecycle Functions - executed when the component instance is removed from the page node tree
relations	Object	No	For definitions of inter-component relationships, see Inter-component Relationships
lifetimes	Object	No	Component Lifecycle Declaration Object, see Component Lifecycle
pageLifetimes	Object	No	The lifecycle declaration object for the page on which the component resides, see Component Lifecycle

©2013-2025 Tencent Cloud International Pte. Ltd.

Code example:

```
// my-behavior.js
module.exports = Behavior({
  behaviors: [],
  properties: {
    myBehaviorProperty: {
      type: String
    }
  },
  data: {
    myBehaviorData: {}
  },
  attached: function(){},
  methods: {
    myBehaviorMethod: function(){}
  }
})
```

Modularity

Last updated：2024-11-28 10:37:01

require

Introduces the module. Returns the interface exposed by the module via module.exports or exports.
If you need to introduce a module from another subpackage, you can configure the callback function to fetch the module asynchronously. If the asynchronous fetch fails, the error callback function will be triggered.

parameter

Name	type	mandatory field	Introductions	Note
path	string	yes	You need to introduce the relative path of the module file relative to the current file, or the npm module name, or npm modules path. Absolute path is not supported	You need to introduce a relative path to the module file relative to the current file, or the npm module name, or the npm module path. Absolute paths are not supported by default, you can customise the path mapping by configuring resolveAlias.
callback	function	no	Asynchronous loading success callback function, the callback function parameters for the successful loading of the module	Asynchronous loading success callback function, the callback function parameter is the successfully loaded module.
error	function	no	Asynchronous load failure callback function, the parameters of this callback function are the error message and module name	Asynchronous load failure callback function, the parameters of this callback function are error message and module name.

require.async Chained calls

It can be used by chaining calls.

```
require
    .async('path/to/mod')
    .then((mod) => {
        console.log(mod)
    })
    .catch(({ errMsg, mod }) => {
```

```
        console.error(`path: ${mod}, ${errMsg}`)
    })
```

Sample Code

Calls within the same package

```
// common.js
function sayHello(name) {
  console.log(`Hello ${name} !`)
}
function sayGoodbye(name) {
  console.log(`Goodbye ${name} !`)
}


module.exports.sayHello = sayHello
exports.sayGoodbye = sayGoodbye
```

```
var common = require('common.js')
Page({
  helloMINA: function() {
    common.sayHello('MINA')
  },
  goodbyeMINA: function() {
    common.sayGoodbye('MINA')
  }
})
```

Asynchronous Calls Across Subpackages

```
// subpackage/common.js Subcontracting common files
export const sayHello = () => console.log("hello")
```

```
// pages/index.js main package page
let common;
require('../../subpackage/common.js', (mod) => {
    common = mod
}, ({ errMsg, mod }) => {
    console.error(`path: ${mod}, ${errMsg}`)
})

Page({
    sayHello() {
```

```
        common && common.sayHello()
    }
})
```

module

Object module

Current Module Object.

attribute

attribute	type	Introductions
exports	Object	Module exposes objects using therequireWhen referencing a module, you can get the

sample code

```
// common.js
function sayHello(name) {
  console.log(`Hello ${name} !`)
}
function sayGoodbye(name) {
  console.log(`Goodbye ${name} !`)
}


module.exports.sayHello = sayHello
exports.sayGoodbye = sayGoodbye
```

exports

Object exports

module.exports A reference to

sample code

```
// common.js
function sayHello(name) {
  console.log(`Hello ${name} !`)
```

```
}
function sayGoodbye(name) {
  console.log(`Goodbye ${name} !`)
}


module.exports.sayHello = sayHello
exports.sayGoodbye = sayGoodbye
```

Basic Function

Last updated：2024-11-28 10:37:30

console

Print the log to the debug panel. console Is a global object that can be accessed directly. In the WeChat client, to vConsole Output log in.

method

console.debug()
Print to Debug Panel debug journal;
console.log()
Print to Debug Panel log journal;
console.info()
Print to Debug Panel info journal;
console.warn()
Print to Debug Panel warn journal;
console.error()
Print to Debug Panel error journal;
console.group(string label)
Create a new grouping in the debug panel. Any subsequent output is indented to indicate that it belongs to the current group. call console.groupEnd after the group;
console.groupEnd()
Ending by console.group Group created.

Be careful

Due to the limited functionality of vConsole and the variations in support for console methods across clients, it is recommended that developers only use the methods provided in this document in their mini programs.

Timer

setTimeout

This API uses number setTimeout(function callback, number delay, any rest)

Function: Set a timer. Execute the registered callback function after the timer expires.
Parameters and description:
function callback, callback function;
number delay, the delay time, the function will be called after the delay, in ms;
any rest, param1, param2, ... , paramN, etc., which are passed as arguments to the callback function.
Return value: number, the number of the timer, this value can be passed to clearTimeout to cancel the timer.

clearTimeout

> The API method is clearTimeout(number timeoutID).

Function: cancels the timer set by setTimeout.
Arguments and description: number timeoutID, the ID of the timer to be cancelled.

setInterval

> The API uses number setInterval(function callback, number delay, any rest)

Function: Set a timer. Execute the registered callback function according to the specified period (in milliseconds).
Parameters and description:
function callback, the callback function;
number delay, the time interval between the execution of the callback function, in ms;
any rest, param1, param2, ... , paramN and so on, which will be passed as parameters to the callback function.
Return value: number, the number of the timer, this value can be passed to clearInterval to cancel the timer.

clearInterval

> The API method is clearInterval(number intervalID)

Function: cancels the timer set by setInterval.
Parameters and description: number intervalID, the ID of the timer to be cancelled.

©2013-2025 Tencent Cloud International Pte. Ltd.

WXML Syntax Reference

Introduce

Last updated：2024-11-21 18:34:18

WXML is a markup language for framework design. It can be used to build the page structure when combined with API Overview and an event system.

Data Binding

```
<!--wxml-->
<view> {{message}} </view>
```

```
// page.js
Page({
  data: {
    message: 'Hello MINA!'
  }
})
```

List Rendering

```
<!--wxml-->
<view wx:for="{{array}}"> {{item}} </view>
```

```
// page.js
Page({
  data: {
    array: [1, 2, 3, 4, 5]
  }
})
```

Conditional Rendering

```
<!--wxml-->
<view wx:if="{{view == 'WEBVIEW'}}"> WEBVIEW </view>
<view wx:elif="{{view == 'APP'}}"> APP </view>
<view wx:else="{{view == 'MINA'}}"> MINA </view>
```

```
// page.js
Page({
  data: {
    view: 'MINA'
  }
})
```

Template

```
<!--wxml-->
<template name="staffName">
  <view>
    FirstName: {{firstName}}, LastName: {{lastName}}
  </view>
</template>

<template is="staffName" data="{{...staffA}}"></template>
<template is="staffName" data="{{...staffB}}"></template>
<template is="staffName" data="{{...staffC}}"></template>
```

```
// page.js
Page({
  data: {
    staffA: {firstName: 'Hulk', lastName: 'Hu'},
    staffB: {firstName: 'Shang', lastName: 'You'},
    staffC: {firstName: 'Gideon', lastName: 'Lin'}
  }
})
```

Data Binding

Last updated：2024-11-21 18:34:18

The dynamic data in WXML comes from the data of the corresponding Page.

Simple Binding

Data binding uses the Mustache syntax (double curly brackets) to wrap the variables. It can be used for the following:

Content

```
<view> {{ message }} </view>
```

```
Page({
  data: {
    message: 'Hello MINA!'
  }
})
```

Component Properties (must be enclosed in double quotes)

```
<view id="item-{{id}}"> </view>
```

```
Page({
  data: {
    id: 0
  }
})
```

Control Properties (must be enclosed in double quotes)

```
<view wx:if="{{condition}}"> </view>
```

```
Page({
  data: {
    condition: true
  }
})
```

©2013-2025 Tencent Cloud International Pte. Ltd.

Keywords (must be enclosed in double quotes)

True: Boolean-type true.

false: Boolean-type false.

```
<checkbox checked="{{false}}"> </checkbox>
```

Note：

Do not directly write **checked="false"**. Its computing result is a string which represents a true value when converted to a Boolean value.

Operations

You can implement simple operations in {{}}. This syntax supports the following methods:

Ternary Operations

```
<view hidden="{{flag ? true : false}}"> Hidden </view>
```

Arithmetic Operations

```
<view> {{a + b}} + {{c}} + d </view>
```

```
Page({
  data: {
    a: 1,
    b: 2,
    c: 3
  }
})
```

The content in view is 3 + 3 + d.

Logical Judgments

```
<view wx:if="{{length > 5}}"> </view>
```

String Operations

```
<view>{{"hello" + name}}</view>
```

```
Page({
  data:{
    name: 'MINA'
  }
})
```

Data path Operations

```
<view>{{object.key}} {{array[0]}}</view>
```

```
Page({
  data: {
    object: {
      key: 'Hello '
    },
    array: ['MINA']
  }
})
```

Combinations

You can also directly implement combinations in Mustache syntax to build new objects and arrays.

Arrays

```
<view wx:for="{{[zero, 1, 2, 3, 4]}}"> {{item}} </view>
```

```
Page({
  data: {
    zero: 0
  }
})
```

The end result is the array [0, 1, 2, 3, 4].

Objects

```
<template is="objectCombine" data="{{for: a, bar: b}}"></template>
```

```
Page({
```

```
  data: {
    a: 1,
    b: 2
  }
})
```

The end result is the object {for: 1, bar: 2}.

You can also use the extended operator ... to extend the object.

```
<template is="objectCombine" data="{{...obj1, ...obj2, e: 5}}"></template>
```

```
Page({
  data: {
    obj1: {
      a: 1,
      b: 2
    },
    obj2: {
      c: 3,
      d: 4
    }
  }
})
```

The end result is the object {a: 1, b: 2, c: 3, d: 4, e: 5}.

If the object's key and value are identical, this can be indirectly expressed.

```
<template is="objectCombine" data="{{foo, bar}}"></template>
```

```
Page({
  data: {
    foo: 'my-foo',
    bar: 'my-bar'
  }
})
```

The end result is the object {foo: 'my-foo', bar:'my-bar'}.

Any combination of the above methods is allowed, but if there are cases where identical variable names occur, the latter will overwrite the former, such as:

```
<template is="objectCombine" data="{{...obj1, ...obj2, a, c: 6}}"></template>
```

```
Page({
  data: {
    obj1: {
```

```
      a: 1,
      b: 2
    },
    obj2: {
      b: 3,
      c: 4
    },
    a: 5
  }
})
```

The end result is the object {a: 5, b: 3, c: 6}.

Note:

If there are spaces between curly brackets and quotes, the content is ultimately parsed into a string.

```
<view wx:for="{{[1,2,3]}} ">
  {{item}}
</view>
```

equivalent to

```
<view wx:for="{{[1,2,3] + ' '}}">
  {{item}}
</view>
```

©2013-2025 Tencent Cloud International Pte. Ltd.

List Rendering

Last updated：2024-11-21 18:34:18

wx:for

In a component, by binding an array using the wx:for control property, you can use the data of the array items to re-render this component.
The subscript variable name of the current item of the default array defaults to index, and the variable name of the current item of the array defaults to item.

```
<view wx:for="{{array}}">
  {{index}}: {{item.message}}
</view>
```

```
Page({
  data: {
    array: [{
      message: 'foo',
    }, {
      message: 'bar'
    }]
  }
})
```

You can use wx:for-item to specify the variable name of the array's current element.
You can use wx:for-index to specify the variable name of the array's current subscript:

```
<view wx:for="{{array}}" wx:for-index="idx" wx:for-item="itemName">
  {{idx}}: {{itemName.message}}
</view>
```

wx:for can also be embedded. Below is the Chinese multiplication table:

```
<view wx:for="{{[1, 2, 3, 4, 5, 6, 7, 8, 9]}}" wx:for-item="i">
  <view wx:for="{{[1, 2, 3, 4, 5, 6, 7, 8, 9]}}" wx:for-item="j">
    <view wx:if="{{i <= j}}">
      {{i}} * {{j}} = {{i * j}}
    </view>
  </view>
</view>
```

©2013-2025 Tencent Cloud International Pte. Ltd.

block wx:for

Similar to block wx:if, you can use wx:for on a <block/> tag to render a structural block containing multiple nodes. For example:

```
<block wx:for="{{[1, 2, 3]}}">
  <view> {{index}}: </view>
  <view> {{item}} </view>
</block>
```

wx:key

If the positions of list items dynamically change or new items are added to the list and you want the items in the list to retain their features and statuses (such as the content input in input and the selection status of switch), you must use wx:key to specify the unique identifiers of the items in the list.
The wx:key is provided in two formats:
String: Represents a property of an item in the for loop array. The value of this property must be a unique string or number in the list and cannot dynamically change.
Reserved keyword `*this` : Represents the item itself in the for loop. This expresses that the item itself is a unique string or number. For example:
When a data change triggers re-rending at the rendering layer, the components that have keys are corrected. The framework ensures that they are reordered, rather than recreated. This ensures the components retain their statuses and improves the efficiency of list rendering.
If wx:key, is not provided, a warning is reported. If you know that this list is static or the order is not important, you can ignore the warning.

Sample code

```
<switch wx:for="{{objectArray}}" wx:key="unique" style="display: block;"> {{item.id
<button bindtap="switch"> Switch </button>
<button bindtap="addToFront"> Add to the front </button>

<switch wx:for="{{numberArray}}" wx:key="*this" style="display: block;"> {{item}} <
<button bindtap="addNumberToFront"> Add to the front </button>
```

```
Page({
  data: {
```

```
    objectArray: [
      {id: 5, unique: 'unique_5'},
      {id: 4, unique: 'unique_4'},
      {id: 3, unique: 'unique_3'},
      {id: 2, unique: 'unique_2'},
      {id: 1, unique: 'unique_1'},
      {id: 0, unique: 'unique_0'},
    ],
    numberArray: [1, 2, 3, 4]
  },
  switch: function(e) {
    const length = this.data.objectArray.length
    for (let i = 0; i < length; ++i) {
      const x = Math.floor(Math.random() * length)
      const y = Math.floor(Math.random() * length)
      const temp = this.data.objectArray[x]
      this.data.objectArray[x] = this.data.objectArray[y]
      this.data.objectArray[y] = temp
    }
    this.setData({
      objectArray: this.data.objectArray
    })
  },
  addToFront: function(e) {
    const length = this.data.objectArray.length
    this.data.objectArray = [{id: length, unique: 'unique_' + length}].concat(this.
    this.setData({
      objectArray: this.data.objectArray
    })
  },
  addNumberToFront: function(e){
    this.data.numberArray = [ this.data.numberArray.length + 1 ].concat(this.data.n
    this.setData({
      numberArray: this.data.numberArray
    })
  }
})
```

Note

When the wx:for value is a string, the string is parsed into a string array.

```
<view wx:for="array">
  {{item}}
```

```
</view>
```

equivalent to

```
<view wx:for="{{['a','r','r','a','y']}}">
  {{item}}
</view>
```

Note:

If there are spaces between curly brackets and quotes, the content is ultimately parsed into a string.

```
<view wx:for="{{[1,2,3]}} ">
  {{item}}
</view>
```

equivalent to

```
<view wx:for="{{[1,2,3] + ' '}}" >
  {{item}}
</view>
```

©2013-2025 Tencent Cloud International Pte. Ltd.

Conditional Rendering

Last updated：2024-11-21 18:34:18

wx:if

In the framework, use `wx:if=""` to determine whether or not to render this code block:

```
<view wx:if="{{condition}}"> True </view>
```

You can also use `wx:elif` and `wx:else` to add an “else” block:

```
<view wx:if="{{length > 5}}"> 1 </view>
<view wx:elif="{{length > 2}}"> 2 </view>
<view wx:else> 3 </view>
```

block wx:if

Because `wx:if` is a control property, you must add it to a tag. To judge multiple component tags at once, you can use a `<block/>` tag to package multiple components together and use the wx:if control property above.

```
<block wx:if="{{true}}">
  <view> view1 </view>
  <view> view2 </view>
</block>
```

Note:

`<block/>` is not a component, just a package element. It does not perform any rendering on the page and only accepts control properties.

wx:if vs hidden

Because the template in `wx:if` can also contain data bindings, when the `wx:if` condition value changes, the framework has a local rendering process that will ensure the conditional block is destroyed or re-rendered when the value changes.

At the same time, `wx:if` is also inert. If the condition is false during initial rendering, the framework does not render it. When the condition changes to true for the first time, local rendering starts.

©2013-2025 Tencent Cloud International Pte. Ltd.

In contrast, hidden is much simpler. The component is always rendered and this property simply controls whether the content is displayed or hidden.
Generally, `wx:if` has higher switching costs, while hidden has higher initial rendering costs. Therefore, in situations with frequent condition changes, it is better to use hidden. If conditions are unlikely to change during runtime, `wx:if` is the better choice.

©2013-2025 Tencent Cloud International Pte. Ltd.

Templates

Last updated：2024-11-21 18:34:18

WXML provides templates, in which you can define code snippets and then call them in different places.

Defining a Template

Use the name property to specify the name of the `template`. Then, define a code snippet in <template/>. For example:

```
<!--
  index: int
  msg: string
  time: string
-->
<template name="msgItem">
  <view>
    <text> {{index}}: {{msg}} </text>
    <text> Time: {{time}} </text>
  </view>
</template>
```

Using the Template

Using the is attribute, declare the template you want to use, and then pass in the data the template needs, for example:

```
<template is="msgItem" data="{{...item}}"/>
```

```
Page({
  data: {
    item: {
      index: 0,
      msg: 'this is a template',
      time: '2016-09-15'
    }
  }
})
```

The `is` property can use Mustache syntax to dynamically determine the template to be rendered:

```
<template name="odd">
  <view> odd </view>
</template>
<template name="even">
  <view> even </view>
</template>

<block wx:for="{{[1, 2, 3, 4, 5]}}">
  <template is="{{item % 2 == 0 ? 'even' : 'odd'}}"/>
</block>
```

Template Scope

Templates have their own scopes. They can only use the data passed by data and the <wxs /> module defined in the file defined by the template.

Referencing

Last updated：2024-11-21 18:34:18

WXML provides two referencing methods `import` and `include`.

import

import can use the template defined by the target file in the current file, for example:
A template named item is defined in item.wxml:

```
<!-- item.wxml -->
<template name="item">
  <text>{{text}}</text>
</template>
```

By referencing item.wxml in index.wxml, you can use the item template:

```
<import src="item.wxml"/>
<template is="item" data="{{text: 'forbar'}}"/>
```

import Scope

The concept of “scope” is used for import references. This means that the reference will import the template defined in the target file, but will not import any import template in the target file.
For example: In the case C import B, B import A, the template defined by B can be used in C and the template defined by A can be used in B. However, C cannot use the template defined by A.

```
<!-- A.wxml -->
<template name="A">
  <text> A template </text>
</template>
```

```
<!-- B.wxml -->
<import src="a.wxml"/>
<template name="B">
  <text> B template </text>
</template>
```

```
<!-- C.wxml -->
```

```
<import src="b.wxml"/>
<template is="A"/>  <!-- Error! Can not use tempalte when not import A. -->
<template is="B"/>
```

include

include can introduce the entire code excluding <template/> <wxs/>. This is equivalent to copying the code to the location of include. For example:

```
<!-- index.wxml -->
<include src="header.wxml"/>
<view> body </view>
<include src="footer.wxml"/>
```

```
<!-- header.wxml -->
<view> header </view>
```

```
<!-- footer.wxml -->
<view> footer </view>
```

WXS Syntax Reference

Introduce

Last updated：2024-11-21 18:34:18

WXS is a set of scripting languages for mini programs that, in combination with WXML, can build the structure of a page.

WXS is a different language from JavaScript and has its own syntax, which is not consistent with JavaScript.

WXS Modules
Variables
Annotations
Operators
Statements
Data Types
Base Library

WXS Modules

Last updated：2024-11-21 18:34:18

WXS code can be written in `<wxs>` tags in a wxml file or in files with the extension .wxs.

Modules

Each `.wxs` file or `<wxs>` tag is an independent module.
Each module has an independent scope. This means the variables and functions in a module are private by default and not visible to other modules.
A module can only externally expose its private variables and functions through the `module.exports` implementation.

wxs Files

In Weixin DevTools, right-click to create a `.wxs` file. You can write WXS scripts directly in the file.
Sample code:

```
// /pages/comm.wxs

var foo = "'hello world' from comm.wxs";
var bar = function(d) {
  return d;
}
module.exports = {
  foo: foo,
  bar: bar
};
```

In the above example, WXS code is written in the /**pages**/**comm.wxs** file. This **.wxs** file can be referenced by other **.wxs** files or **<wxs>** tags in WXML.

module Objects

Each **`wxs`** module contains one built-in `module` object.

Properties

exports: This property is used to share the module's private variables and functions externally.
Sample code:

```
// /pages/tools.wxs
```

```
var foo = "'hello world' from tools.wxs";
var bar = function (d) {
  return d;
}
module.exports = {
  FOO: foo,
  bar: bar,
};
module.exports.msg = "some msg";
```

```
<!-- page/index/index.wxml -->

<wxs src="./../tools.wxs" module="tools" />
<view> {{tools.msg}} </view>
<view> {{tools.bar(tools.FOO)}} </view>
```

Page output:

```
some msg
'hello world' from tools.wxs
```

require Functions

You can use a `require` function to reference another `wxs` file module in a `.wxs` module.

When referencing another `wxs` file module, you must note the following:

You can only reference `.wxs` files modules and must use relative paths;

All `wxs` modules are single instances. The first time a wxs module is referenced, it is automatically initialized as a single-instance object. Multiple pages, multiple regions, and multiple references all use the same wxs module object;

If a wxs module is never referenced after it is defined, the module is not parsed or run.

Sample code:

```
// /pages/tools.wxs

var foo = "'hello world' from tools.wxs";
var bar = function (d) {
  return d;
}
module.exports = {
  FOO: foo,
  bar: bar,
};
module.exports.msg = "some msg";
```

```
// /pages/logic.wxs
```

```
var tools = require("./tools.wxs");

console.log(tools.FOO);
console.log(tools.bar("logic.wxs"));
console.log(tools.msg);
```

```
<!-- /page/index/index.wxml -->

<wxs src="./../logic.wxs" module="logic" />
```

Console output:

```
'hello world' from tools.wxs
logic.wxs
some msg
```

<wxs> Tags

Property Name	Type	Default Value	Description
module	String	-	The module name of the current <wxs> tag. This is a required field.
src	String	-	The relative path of the referenced .wxs file, only valid when this tag is a single closed tag or the tag content is empty.

module Property

The module property is the module name of the current `<wxs>` tag. In a single wxml file, we recommend that this value be unique. In the case of duplicate module names, the value is overwritten based on the order (the latter overwrites the former). The names of wxs modules are not overwritten across different files.

The name specified by the module property value must comply with the following rules:

The first character must be an English letter (a-z, A-Z) or underscore (_).

The remaining characters can be English letters (a-z, A-Z), underscores (_), or numbers (0-9).

Sample code:

```
<!--wxml-->

<wxs module="foo">
var some_msg = "hello world";
module.exports = {
  msg : some_msg,
```

```
}
</wxs>
<view> {{foo.msg}} </view>
```

Page output:

```
hello world
```

The above example declares a module with the name foo that exposes the some_msg variable for use on the current page.

src Property

The src property is used to reference another wxs file module.
When referencing another wxs file module, you must note the following:
You can only reference .wxs files modules and must use relative paths.
All wxs modules are single instances. The first time a wxs module is referenced, it is automatically initialized as a single-instance object. Multiple pages, multiple regions, and multiple references all use the same wxs module object.
If a wxs module is never referenced after it is defined, the module is not parsed or run.
Sample code:

```
// /pages/index/index.js

Page({
  data: {
    msg: "'hello wrold' from js",
  }
})
```

```
<!-- /pages/index/index.wxml -->

<wxs src="./../comm.wxs" module="some_comms"></wxs>
<!-- You can also directly use single tag closed syntax.
<wxs src="./../comm.wxs" module="some_comms" />
-->

<!-- Calls the bar function in the some_comms module, and the parameter is foo in t
<view> {{some_comms.bar(some_comms.foo)}} </view>
<!-- Calls the bar function in the some_comms module, and the parameter is msg in p
<view> {{some_comms.bar(msg)}} </view>
```

Page output:

```
'hello world' from comm.wxs
'hello wrold' from js
```

In the above example, the <wxs> tag is used to reference /page/comm.wxs in the file /page/index/index.wxml.

Note

A `<wxs>` module can only be accessed in the WXML file that defines the module. If you use `<include>` or `<import>`, `<wxs>` modules are not introduced into the corresponding WXML files.
In a `<template>` tag, you can only use the `<wxs>` module defined in the WXML file that defines the `<template>`.

©2013-2025 Tencent Cloud International Pte. Ltd.

Variables

Last updated：2025-03-25 11:41:05

Concepts

Variables in WXS are references to values;
Variables that are not declared and used by direct assignment are defined as global variables;
If a variable is declared without being assigned a value, the default value is undefined;
var behaves in the same way as javascript, there will be variable lifting.

```
var foo = 1;
var bar = "hello world";
var i; // i === undefined
```

In the above code, three variables foo, bar and i are declared. Then, foo is assigned the value 1 and bar is assigned the string "hello world".

Variable Naming

Variable naming must conform to the following two rules:
The first character must be: letter (a-zA-Z), underscore (_);
The remaining characters can be: letters (a-zA-Z), underscore (_), numbers (0-9).

Reserved Identifier

The following identifiers cannot be used as variable names:

```
delete
void
typeof

null
undefined
NaN
Infinity
var

if
```

```
else

true
false

require

this
function
arguments
return

for
while
do
break
continue
switch
case
default
```

Annotations

Last updated：2024-11-21 18:34:18

WXS uses three main annotation methods.

Sample code:

```
<!-- wxml -->
<wxs module="sample">
// Method 1: Single-line annotations

/*
Method 2: Multi-line annotations
*/

/*
Method 3: End annotations. This annotates all the  WXS  code starting from  /* .

var a = 1;
var b = 2;
var c = "fake";

</wxs>
```

In the above example, all WXS code is annotated.

Note:

The only difference between method 3 and method 2 is the lack of the */ terminator.

Operators

Last updated：2024-11-21 18:34:19

Basic Operators

Sample code:

```
var a = 10, b = 20;

// Addition
console.log(30 === a + b);
// Subtraction
console.log(-10 === a - b);
// Multiplication
console.log(200 === a * b);
// Division
console.log(0.5 === a / b);
// Complementation
console.log(10 === a % b);
```

The addition operation (+) can also be used to concatenate strings.

```
var a = '.w' , b = 'xs';

// String concatenation
console.log('.wxs' === a + b);
```

Unary Operators

Sample code:

```
var a = 10, b = 20;

// Auto increment
console.log(10 === a++);
console.log(12 === ++a);
// Auto decrement
console.log(12 === a--);
console.log(10 === --a);
// Positive value
console.log(10 === +a);
```

```
// Negative value
console.log(0-10 === -a);
// Not
console.log(-11 === ~a);
// Negate
console.log(false === !a);
// Delete
console.log(true === delete a.fake);
// Void
console.log(undefined === void a);
// Typeof
console.log("number" === typeof a);
```

Bit Operators

Sample code:

```
var a = 10, b = 20;

// Left shift
console.log(80 === (a << 3));
// Unsigned right shift
console.log(2 === (a >> 2));
// Signed right shift
console.log(2 === (a >>> 2));
// AND
console.log(2 === (a & 3));
// XOR
console.log(9 === (a ^ 3));
// OR
console.log(11 === (a | 3));
```

Relational Operators

Sample code:

```
var a = 10, b = 20;

// Less than
console.log(true === (a < b));
// Greater than
console.log(false === (a > b));
```

```
// Less than or equal to
console.log(true === (a <= b));
// Greater than or equal to
console.log(false === (a >= b));
```

Equivalence Operators

Sample code:

```
var a = 10, b = 20;

// Equal sign
console.log(false === (a == b));
// Not equal sign
console.log(true === (a != b));
// Equivalent sign
console.log(false === (a === b));
// Not equivalent sign
console.log(true === (a !== b));
```

Assignment Operators

Sample code:

```
var a = 10;

a = 10; a *= 10;
console.log(100 === a);
a = 10; a /= 5;
console.log(2 === a);
a = 10; a %= 7;
console.log(3 === a);
a = 10; a += 5;
console.log(15 === a);
a = 10; a -= 11;
console.log(-1 === a);
a = 10; a <<= 10;
console.log(10240 === a);
a = 10; a >>= 2;
console.log(2 === a);
a = 10; a >>>= 2;
console.log(2 === a);
```

```
a = 10; a &= 3;
console.log(2 === a);
a = 10; a ^= 3;
console.log(9 === a);
a = 10; a |= 3;
console.log(11 === a);
```

Binary Logical Operators

Sample code:

```
var a = 10, b = 20;

// Logical AND
console.log(20 === (a && b));
// Logical OR
console.log(10 === (a || b));
```

Other Operators

Sample code:

```
var a = 10, b = 20;

//Conditional operator
console.log(20 === (a >= 10 ? a + 10 : b + 10));
//Comma operator
console.log(20 === (a, b));
```

Operator Priority

Priority	Operator	Description	Associativity
20	(...)	Brackets	n/a
19	... .	Member access	Left to right
	... [...]	Member access	Left to right
	... (...)	Function call	Left to right

17	... ++	Postfix increment	n/a
	... --	Postfix decrement	n/a
16	! ...	Logical not	Right to left
	~ ...	Bitwise not	Right to left
	+ ...	Unary addition	Right to left
	- ...	Unary subtraction	Right to left
	++ ...	Prefix increment	Right to left
	-- ...	Prefix decrement	Right to left
	typeof ...	typeof	Right to left
	void ...	void	Right to left
	delete ...	delete	Right to left
14	... * ...	Multiplication	Left to right
	... / ...	Division	Left to right
	... % ...	Modulo	Left to right
13	... + ...	Addition	Left to right
	... - ...	Subtraction	left to right
12	... << ...	Bitwise left shift	Left to right
	... >> ...	Bitwise right shift	Left to right
	... >>> ...	Unsigned right shift	Left to right
11	... < ...	Less than	Left to right
	... <= ...	Less than or equal to	Left to right
	... > ...	Greater than	Left to right
	... >= ...	Greater than or equal to	Left to right
10	... == ...	Equal sign	Left to right
	... != ...	Not equal sign	Left to right

	... === ...	Equivalent sign	Left to right
	... !== ...	Not equivalent sign	Left to right
9	... & ...	Bitwise AND	Left to right
8	... ^ ...	Bitwise XOR	Left to right
7	... | ...	Bitwise OR	Left to right
6	... && ...	Logical AND	Left to right
5	... || ...	Local OR	Left to right
4	... ? ... : ...	Conditional operator	Left to right
3	... = ...	Value assignment	Left to right
	... += ...	Value assignment	Left to right
	... -= ...	Value assignment	Left to right
	... *= ...	Value assignment	Left to right
	... /= ...	Value assignment	Left to right
	... %= ...	Value assignment	Left to right
	... <<= ...	Value assignment	Left to right
	... >>= ...	Value assignment	Left to right
	... >>>= ...	Value assignment	Left to right
	... &= ...	Value assignment	Left to right
	... ^= ...	Value assignment	Left to right
	... |= ...	Value assignment	Right to left
0	... , ...	Comma	Left to right

Statements

Last updated：2024-11-21 18:34:18

if Statements

In WXS, you can use if statements in the following formats:
if (expression) statement : Execute statement when expression is truthy.
if (expression) statement1 else statement2 : Execute statement1 when expression is truthy. Otherwise, execute statement2.
if ... else if ... else statementN Using this statement, you can choose one of the executions from statement1 to statementN.
Example Syntax:

```
// if ...

if (Expression) Statement;

if (Expression)
  Statement;

if (Expression) {
  Code block;
}


// if ... else

if (Expression) Statement;
else Statement;

if (Expression)
  Statement;
else
  Statement;

if (Expression) {
  Code block;
} else {
  Code block;
}

// if ... else if ... else ...
```

```
if (Expression) {
  Code block;
} else if (Expression) {
  Code block;
} else if (Expression) {
  Code block;
} else {
  Code block;
}
```

switch Statements

Example Syntax:

```
switch (Expression) {
  case Variable:
    Statement;
  case Number:
    Statement;
    break;
  case String:
    Statement;
  default:
    Statement;
}
```

default branches can be omitted.

After the case keyword, you can only use a variable, number, or string.

Sample code:

```
var exp = 10;

switch ( exp ) {
case "10":
  console.log("string 10");
  break;
case 10:
  console.log("number 10");
  break;
case exp:
  console.log("var exp");
  break;
default:
```

```
  console.log("default");
}
```

Output:

```
number 10
```

for Statements

Example Syntax:

```
for (Statement; Statement; Statement)
  Statement;

for (Statement; Statement; Statement) {
  Code block;
}
```

Supports the use of break and continue keywords.

Sample code:

```
for (var i = 0; i < 3; ++i) {
  console.log(i);
  if( i >= 1) break;
}
```

Output:

```
0
1
```

while Statements

Example Syntax:

```
while (Expression)
  Statement;

while (Expression){
  Code block;
}
```

```
do {
  Code block;
} while (Expression)
```

When expression is true, loop execute statement or code block.

Supports the use of break and continue keywords.

Data Types

Last updated：2025-03-17 17:23:30

The WXS language currently supports the following data types:

number;

string;

boolean;

object;

function;

array;

date;

regexp;

number

Syntax

There are two types of numbers: integers and decimals.

```
var a = 10;
var PI = 3.141592653589793;
```

Properties

constructor: Returns the string "Number"

Methods

`toString` ;

`toLocaleString` ;

`valueOf` ;

`toFixed` ;

`toExponential` ;

`toPrecision` 。

For information on using the above methods, refer to the ES5 standard.

©2013-2025 Tencent Cloud International Pte. Ltd.

string

Syntax

here are two ways to write a string:

```
'hello world';
"hello world";
```

Properties

`constructor` : Returns the string "String";

`length` 。

> For explanations of properties other than constructor, refer to the ES5 standard.

Methods

`toString` ;

`valueOf` ;

`charAt` ;

`charCodeAt` ;

`concat` ;

`indexOf` ;

`lastIndexOf` ;

`localeCompare` ;

`match` ;

`replace` ;

`search` ;

`slice` ;

`split` ;

`substring` ;

`toLowerCase` ;

`toLocaleLowerCase` ;

`toUpperCase` ;

`toLocaleUpperCase` ;

`trim` 。

For information on using the above methods, refer to the ES5 standard.

boolean

Syntax

Boolean values can take one of two values: true or false.

Properties

`constructor` : Returns the string "Boolean".

Methods

`toString` ;

`valueOf` 。

For information on using the above methods, refer to the ES5 standard.

object

Syntax

An object is a type of unordered key-value pair, which can be used in the following ways:

```
var o = {} // Generates a new empty object.

// Generates a new non-empty object.
o = {
  'string'  : 1,  //The object's key can be a string.
  const_var : 2,  //The object's key can also be an identifier that conforms to the
  func      : {}, //The object's value can be of any type.
};

//Object property read operation
console.log(1 === o['string']);
console.log(2 === o.const_var);

//Object property write operation
o['string']++;
o['string'] += 10;
```

```
o.const_var++;
o.const_var += 10;

//Object property read operation
console.log(12 === o['string']);
console.log(13 === o.const_var);
```

Properties

constructor: Returns the string "Object".

```
console.log("Object" === {k:"1",v:"2"}.constructor)
```

Methods

toString: Returns the string "[object Object]".

function

Syntax

function supports the following definition methods:

```
//Method 1
function a (x) {
  return x;
}

//Method 2
var b = function (x) {
  return x;
}
```

function also supports the following syntax (anonymous functions, closures, etc.):

```
var a = function (x) {
  return function () { return x;}
}

var b = a(100);
console.log( 100 === b() );
```

arguments

In function, you can use the arguments keyword. This keyword only has the following properties at present:

length: The number of arguments passed to the function.

[index]: Each argument passed to the function can be traversed by the index subscript.

Sample code:

```
var a = function(){
  console.log(3 === arguments.length);
  console.log(100 === arguments[0]);
  console.log(200 === arguments[1]);
  console.log(300 === arguments[2]);
};
a(100,200,300);
```

Properties

constructor: Returns the string "Function".

length: Returns the number of formal parameters of the function

Methods

toString: Returns the string "[function Function]".

Sample code:

```
var func = function (a,b,c) { }

console.log("Function" === func.constructor);
console.log(3 === func.length);
console.log("[function Function]" === func.toString());
```

array

Syntax

array supports the following definition methods:

```
var a = [];       // Generates a new empty array.

a = [1,"2",{},function(){}];  // Generates a new non-empty array, with array elemen
```

Properties

`constructor` : Returns the string "Array";

`length` 。

For explanations of properties other than constructor, refer to the ES5 standard.

Methods

`toString` ;
`concat` ;
`join` ;
`pop` ;
`push` ;
`reverse` ;
`shift` ;
`slice` ;
`sort` ;
`splice` ;
`unshift` ;
`indexOf` ;
`lastIndexOf` ;
`every` ;
`some` ;
`forEach` ;
`map` ;
`filter` ;
`reduce` ;
`reduceRight` 。

For information on using the above methods, refer to the ES5 standard.

date

Syntax

Generates the getDate function required by the date object and returns a current time object.

```
getDate()
getDate(milliseconds)
getDate(datestring)
```

```
getDate(year, month[, date[, hours[, minutes[, seconds[, milliseconds]]]]])
```

Parameters

milliseconds: The number of milliseconds from 00:00:00 UTC January 1, 1970.

datestring: The date string, in the format: "month day, year hours:minutes:seconds".

Sample code:

```
var date = getDate(); //Returns the current time object.

date = getDate(1500000000000);
// Fri Jul 14 2017 10:40:00 GMT+0800 (China Standard Time)
date = getDate('2017-7-14');
// Fri Jul 14 2017 00:00:00 GMT+0800 (China Standard Time)
date = getDate(2017, 6, 14, 10, 40, 0, 0);
// Fri Jul 14 2017 10:40:00 GMT+0800 (China Standard Time)
```

Properties

constructor: Returns the string "Date".

Methods

`toString` ;

`toDateString` ;

`toTimeString` ;

`toLocaleString` ;

`toLocaleDateString` ;

`toLocaleTimeString` ;

`valueOf` ;

`getTime` ;

`getFullYear` ;

`getUTCFullYear` ;

`getMonth` ;

`getUTCMonth` ;

`getDate` ;

`getUTCDate` ;

`getDay` ;

`getUTCDay` ;

`getHours` ;

`getUTCHours` ;

`getMinutes` ;

©2013-2025 Tencent Cloud International Pte. Ltd.

`getUTCMinutes` ;
`getSeconds` ;
`getUTCSeconds` ;
`getMilliseconds` ;
`getUTCMilliseconds` ;
`getTimezoneOffset` ;
`setTime` ;
`setMilliseconds` ;
`setUTCMilliseconds` ;
`setSeconds` ;
`setUTCSeconds` ;
`setMinutes` ;
`setUTCMinutes` ;
`setHours` ;
`setUTCHours` ;
`setDate` ;
`setUTCDate` ;
`setMonth` ;
`setUTCMonth` ;
`setFullYear` ;
`setUTCFullYear` ;
`toUTCString` ;
`toISOString` ;
`toJSON` 。

For information on using the above methods, refer to the ES5 standard.

regexp

Syntax

Generates the getRegExp function to be used by the regexp object.

```
getRegExp(pattern[, flags])
```

Parameters:

pattern: The content of the regular expression.

flags: The modifiers. This field can only include the following characters:

g: global

i: ignoreCase

m: multiline

Sample code:

```
var a = getRegExp("x", "img");
console.log("x" === a.source);
console.log(true === a.global);
console.log(true === a.ignoreCase);
console.log(true === a.multiline);
```

Properties

`constructor` : Returns the string `"RegExp"` ;

`source` ;

`global` ;

`ignoreCase` ;

`multiline` ;

`lastIndex` 。

For explanations of properties other than constructor, refer to the ES5 standard.

Methods

`exec` ;

`test` ;

`toString` 。

For information on using the above methods, refer to the ES5 standard.

Data Type Determination

constructor Property

You can use the constructor property to determine the data type.

Sample code:

```
var number = 10;
console.log( "Number" === number.constructor );

var string = "str";
console.log( "String" === string.constructor );

var boolean = true;
console.log( "Boolean" === boolean.constructor );

var object = {};
console.log( "Object" === object.constructor );

var func = function(){};
console.log( "Function" === func.constructor );

var array = [];
console.log( "Array" === array.constructor );

var date = getDate();
console.log( "Date" === date.constructor );

var regexp = getRegExp();
console.log( "RegExp" === regexp.constructor );
```

typeof

You can also use typeof to distinguish certain data types.

Sample code:

```
var number = 10;
var boolean = true;
var object = {};
var func = function(){};
var array = [];
var date = getDate();
var regexp = getRegExp();

console.log( 'number' === typeof number );
console.log( 'boolean' === typeof boolean );
console.log( 'object' === typeof object );
console.log( 'function' === typeof func );
console.log( 'object' === typeof array );
console.log( 'object' === typeof date );
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
console.log( 'object' === typeof regexp );

console.log( 'undefined' === typeof undefined );
console.log( 'object' === typeof null );
```

©2013-2025 Tencent Cloud International Pte. Ltd.

Base Library

Last updated：2024-11-21 18:34:18

console

The `console.log` method is used to output messages in the console window. This method can accept multiple parameters and concatenate their results for output.

Math

Properties

`E`
`LN10`
`LN2`
`LOG2E`
`LOG10E`
`PI`
`SQRT1_2`
`SQRT2`

> For information on using the above properties, refer to the ES5 standard.。

Methods

`abs`
`acos`
`asin`
`atan`
`atan2`
`ceil`
`cos`
`exp`
`floor`
`log`

`max`
`min`
`pow`
`random`
`round`
`sin`
`sqrt`
`tan`

For information on using the above properties, refer to the ES5 standard.

JSON

Methods

stringify(object): Converts the `object` object into a `JSON` string and returns the string.

parse(string): Converts the `JSON` string into an object and returns the object.

Sample code:

```
console.log(undefined === JSON.stringify());
console.log(undefined === JSON.stringify(undefined));
console.log("null"===JSON.stringify(null));

console.log("111"===JSON.stringify(111));
console.log('"111"'===JSON.stringify("111"));
console.log("true"===JSON.stringify(true));
console.log(undefined===JSON.stringify(function(){}));


console.log(undefined===JSON.parse(JSON.stringify()));
console.log(undefined===JSON.parse(JSON.stringify(undefined)));
console.log(null===JSON.parse(JSON.stringify(null)));

console.log(111===JSON.parse(JSON.stringify(111)));
console.log("111"===JSON.parse(JSON.stringify("111")));
console.log(true===JSON.parse(JSON.stringify(true)));

console.log(undefined===JSON.parse(JSON.stringify(function(){})));
```

Number

Properties

`MAX_VALUE`

`MIN_VALUE`

`NEGATIVE_INFINITY`

`POSITIVE_INFINITY`

> For information on using the above properties, refer to the ES5 standard.

Date

Properties

`parse`

`UTC`

now

> For information on using the above properties, refer to the ES5 standard.

Global

Properties

`NaN`

`Infinity`

`undefined`

> For information on using the above properties, refer to the ES5 standard.

Methods

`parseInt`

`parseFloat`

`isNaN`

`isFinite`

`decodeURI`

`decodeURIComponent`

`encodeURI`

`encodeURIComponent`

For information on using the above properties, refer to the ES5 standard.

©2013-2025 Tencent Cloud International Pte. Ltd.

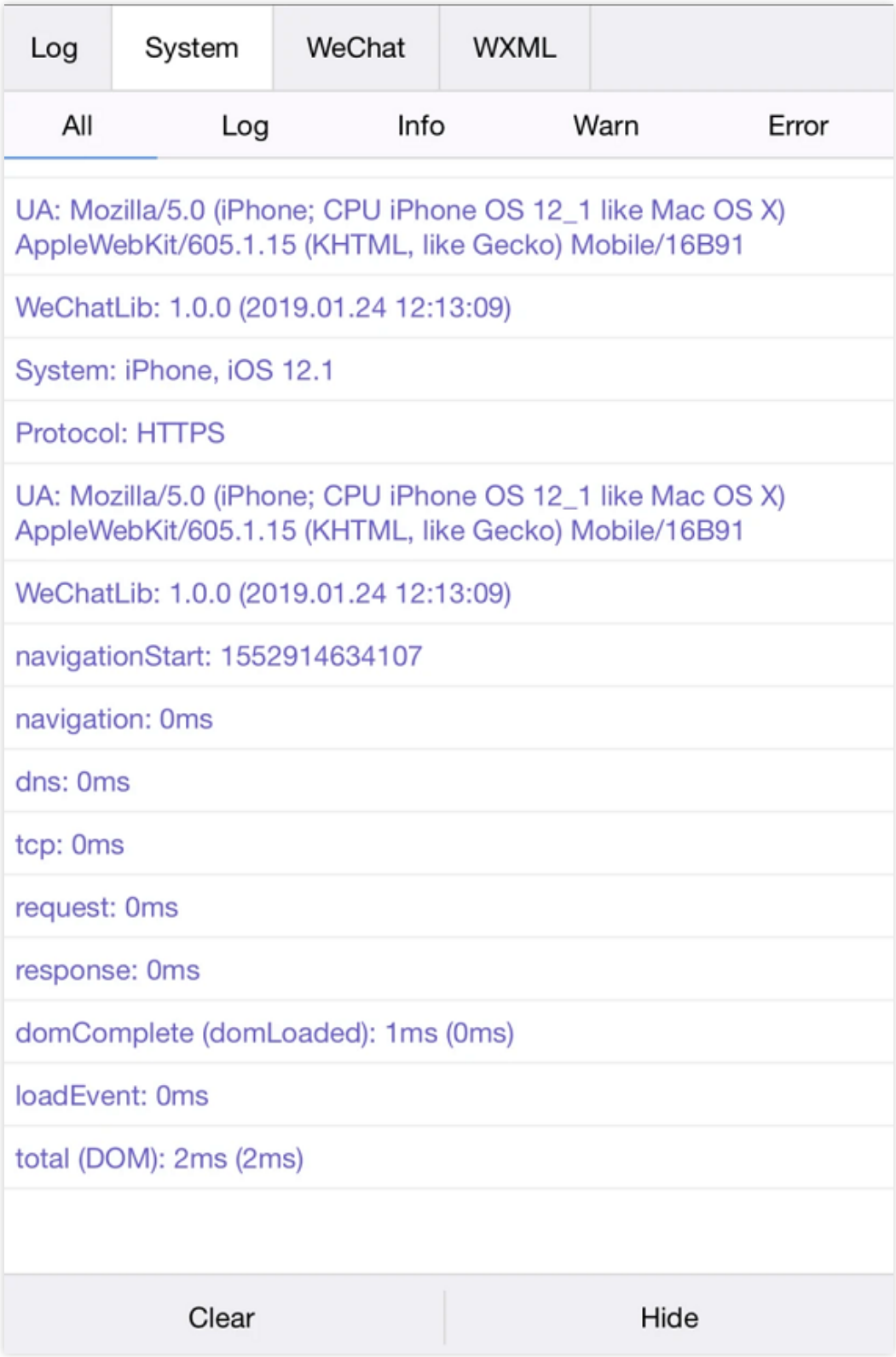
Availability

Last updated：2025-03-17 17:23:30

Debug

vConsole

On the actual device, if you wish to view the log content output by the console API and additional debugging information, select **Open Debugging** from the menu opened by clicking the button in the top right corner of the screen. At this point, the mini program will exit, and upon reopening, a **vConsole** button will appear in the bottom right corner. Clicking the **vConsole** button will open the log panel.
The content displayed by the mini program's vConsole is as follows:



vConsole usage description

Due to the limitations of the implementation mechanism, the log content printed by the developer calling the console API is converted into a JSON string and then transmitted to the `vConsole`. As a result, there are some restrictions on the content displayed in the vConsole:

Except for `Number`, `String`, `Boolean`, and `null`, other types will be processed and displayed as Object, and the Enumerable properties in the object and prototype chain will be printed.
`Infinity` and `NaN` will appear as `null`.
`Undefined`, `ArrayBuffer`, and `Function` types cannot be displayed.
Objects with circular references cannot be printed.

```
const a = {}
a.b = a
console.log(a)
```

In response to the above issues, some measures have been taken when vConsole is used in the mini program.

```
const circular = {x: {}, c: {}}
circular.x = [{promise: Promise.resolve()}]
circular.a = circular
circular.c.x0 = circular.x[0]

console.log(circular)
// "{a: '<Circular: @>', c: {x0: '<Circular: @.x[0]>'}, x: [{promise: '<Promise>'}]
```

Note:
Avoid printing overly complex or lengthy log content in non-debugging scenarios whenever possible, as it may result in additional time consumption.

Runtime Environment

Runtime environment of the mini program

The mini program operates across three platforms: iOS (iPhone/iPad), Android, and the developer tool used for debugging.
The script execution environments across the three platforms, as well as the environment used for rendering non-native components, are distinct from each other:
On iOS, the JavaScript code for the logic layer of the mini program runs within JavaScriptCore, while the view layer is rendered by WKWebView. The environment includes iOS8, iOS9, and iOS10.
On Android, the JavaScript code for the logic layer of the mini program operates within X5 JSCore, while the view layer, based on the Mobile Chrome 97 kernel, implements the function of same-layer rendering.
Within the developer tool, the JavaScript code for the logic layer of the mini program operates within NW.js, while the view layer is rendered by Chromium 60 Webview.

Operational Constraints

For security reasons, dynamic execution of JS code is not supported within the mini program, that is:
The use of `eval` to execute `JS` code is not supported.
The creation of functions using `new Function` is not supported.

Platform differences

Despite the striking similarities among the three platforms, there are still subtle differences:
Inconsistencies in **JavaScript** syntax and API support: Developers can circumvent syntax issues by enabling the `ES6` to `ES5` conversion feature. Additionally, the Mini Program's base library incorporates necessary Polyfills to compensate for API discrepancies.
Inconsistent rendering performance in `WXSS` : Although most issues can be circumvented by enabling style completion, it is still recommended that developers individually inspect the actual performance of the mini program on iOS and Android.
Note:
The developer tools are intended solely for debugging purposes, with the final performance being determined by the client.

Client ES6 API Support Status

Note:
In the following table, √ indicates support, while × signifies no support.

String	iOS 8	iOS9	iOS10	Android
codePointAt	√	√	√	√
normalize	×	×	√	√
includes	√	√	√	√
startsWith	√	√	√	√
endsWith	√	√	√	√
repeat	√	√	√	√
String.fromCodePoint	√	√	√	√

Array	iOS 8	iOS9	iOS10	Android
copyWithin	√	√	√	√
find	√	√	√	√
findIndex	√	√	√	√

fill	√	√	√	√
entries	√	√	√	√
keys	√	√	√	√
values	×	√	√	×
includes	×	√	√	√
Array.from	√	√	√	√
Array.of	√	√	√	√

Number	iOS 8	iOS9	iOS10	Android
isFinite	√	√	√	√
isNaN	√	√	√	√
parseInt	√	√	√	√
parseFloat	√	√	√	√
isInteger	√	√	√	√
EPSILON	√	√	√	√
isSafeInteger	√	√	√	√

Math	iOS 8	iOS9	iOS10	Android
trunc	√	√	√	√
sign	√	√	√	√
cbrt	√	√	√	√
clz32	√	√	√	√
imul	√	√	√	√
fround	√	√	√	√
hypot	√	√	√	√
expm1	√	√	√	√

©2013-2025 Tencent Cloud International Pte. Ltd.

log1p	√	√	√	√
log10	√	√	√	√
log2	√	√	√	√
sinh	√	√	√	√
cosh	√	√	√	√
tanh	√	√	√	√
asinh	√	√	√	√
acosh	√	√	√	√
atanh	√	√	√	√

Object	iOS 8	iOS9	iOS10	Android
is	√	√	√	√
assign	√	√	√	√
getOwnPropertyDescriptor	√	√	√	√
keys	√	√	√	√
getOwnPropertyNames	√	√	√	√
getOwnPropertySymbols	√	√	√	√

Other	iOS 8	iOS9	iOS10	Android
Symbol	√	√	√	√
Set	√	√	√	√
Map	√	√	√	√
Proxy	×	×	√	×

Reflect	√	√	√	√
Promise	√	√	√	√

Operational mechanism

There are two scenarios when a mini program is launched: one is a **cold start**, and the other is a **hot start**. If a user has already opened a certain mini program and then reopens it within a certain period, there is no need for a restart; instead, the mini program in the backend is simply switched to the frontend, a process known as a hot start. A cold start refers to the situation where the user opens the mini program for the first time or reopens it after it has been actively terminated by the host app, in which case the mini program needs to be reloaded and launched.

Update Mechanism

Upon a cold start, if a new version of the mini program is detected, the client will asynchronously download the new version's code package while simultaneously launching with the local package. This means that the new version of the mini program will not be applied until the next cold start. If immediate application of the latest version is required, the wx.getUpdateManager API can be used.

Operational mechanism

The concept of a restart does not apply to mini programs.
When the mini program enters the backend, the client will maintain a running status for a certain period. After this period (currently set at 5 minutes), it will be actively terminated by the container.
On iOS, if the client receives two or more system memory warnings within a certain time interval (currently set at 5 seconds), it will actively destroy the mini program and alert the user that **the mini program may cause the host to respond slowly and has been terminated**. It is recommended to perform necessary memory cleanup.

©2013-2025 Tencent Cloud International Pte. Ltd.

Components

Component Overview

Last updated：2025-03-20 18:01:55

View Container

Name	Feature Description	Minimum Version
scroll-view	Scrollable View Region	1.5.0
swiper	Slider View Container	1.5.0
swiper-item	Can only be placed within the Swiper component, with width and height automatically set to 100%.	1.5.0
view	View Container	1.5.0
movable-area	Movable Area of the Movable-View	1.5.0
movable-view	A movable view container that can be dragged and slid across the page.	1.5.0
cover-image	An image view overlaying the native components.	1.5.0
cover-view	A text view overlaying the native components.	1.5.0

Fundamental Content

Name	Feature Description	Minimum Version
icon	Icon Component	1.5.0
progress	Progress Bar	1.5.0
text	Text	1.5.0
rich-text	rich text	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

Form Component

Name	Feature Description	Minimum Version
button	Button	1.5.0
checkbox	Multiple Selection Items	1.5.0
checkbox-group	Multi-item selector, internally composed of multiple checkboxes.	1.5.0
form	Form	1.5.0
input	Input Box	1.5.0
label	Enhancing the usability of form components.	1.5.0
picker	A scrolling selector that pops up from the bottom.	1.5.0
radio	Single selection item.	1.5.0
radio-group	Single-item selector, composed of multiple radio buttons.	1.5.0
slider	Sliding Selector	1.5.0
switch	Toggle Selector	1.5.0
textarea	Multi-line input field.	1.5.0

Navigation

Name	Feature Description	Minimum Version
navigator	Page Link	1.5.0

Media Component

Name	Feature Description	Minimum Version
image	Image	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

video	Video	1.5.0
animation-video	Transparent Video	1.5.0
camera	System Camera	1.5.0
live-player	Real-Time Audio-Video Playback	1.5.0
live-pusher	Real-Time Audio-Video Recording	1.5.0

Map

Name	Feature Description	Minimum Version
map	Map	1.5.0

canvas

Name	Feature Description	Minimum Version
Canvas	canvas	1.5.0

Open Capabilities

Name	Feature Description	Minimum Version
web-view	Container for Hosting Web Pages	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

Native Components

Last updated：2025-04-22 18:11:25

Native Components

Our certain components are native, created by the client-side. These native components, implemented by the Mini Program client, boast superior performance and a richer array of features.
Native Components include:
camera: A camera component, utilized for invoking the camera to capture photographs or record videos；
Canvas: A canvas component, employed for the creation of graphics, animations, and the like；
input (Manifests as a native component only when in focus)；
textarea: A multiline input box component, designed for the input of multiline text；
video: A video component, designed for video playback；
live-player: A real-time audio and video playback component used for playing live streams；
live-pusher：A real-time audio and video recording component used for pushing live streams；
map: A map component, utilized for the display of maps along with markers, routes, and other features on the map；
ad：Banner advertisement；
web-view：A container that hosts the H5 pages.

Usage restrictions for the native components

Given that native components exist outside of the WebView rendering process, the following limitations apply when using them:
The hierarchy of native components is of the **highest** order. Therefore, regardless of the `z-index` set for other components on the page, they cannot overlay the native components. Subsequently inserted native components can overlay the previously inserted ones.
Native components cannot be used within the `<picker-view>` .
Certain CSS styles cannot be applied to native components, for example:
CSS animations cannot be applied to native components.
Defining native components as `position: fixed` is not possible.
The display area of native components cannot be cropped by using `overflow: hidden` on parent nodes.
Event listeners for native components do not support the `bind:eventname` syntax, only `bindeventname` is supported. Native components also do not support event binding methods such as `catch` and `capture` .
Native components will obscure the debug panel popped up by vConsole.
Note:

On the tool, native components are simulated with web components. Hence, in many cases, they cannot accurately replicate the performance on a real device. It is recommended that developers debug on real devices whenever possible when using native components.

Non-same-layer rendering limitations on Android platform

Native components use same-layer rendering on iOS, while Android is non-same-layer rendering, so there are the following limitations specific to the Android platform:

1. Absolute Layer Priority

Android native components are always at the topmost layer of the view, even after the insertion of ordinary components can not be covered.

Solution:

Use the components cover-view and cover-image which are specially designed for overriding native components.

Example Code

```
<video>
    <cover-view class="controls">Customize Play Control Button</cover-view>
</video>
```

Note：

Only supports nesting within the specified native component.

Limited style attributes (only simple styles such as `opacity\\padding\\border-radius\\rotate\\scaleX\\scaleY\\width\\height\\left\\top\\background-color` , etc. are supported).

Cannot contain other regular child components.

2. Dynamic position adjustment is limited

When dynamically adjusting the position of native components, flickering and shadows may occur, especially when moving and scaling rapidly, the asynchronous rendering mechanism between the native rendering layer and the WebView view layer may lead to screen tearing.

Solution:

Respond to touch events via wxs, calculate coordinates in real time, and dynamically adjust the position of native components. Since wxs can directly handle events from the view layer, it reduces the communication overhead with the logic layer. For example, for frequently triggered touch events, wxs can be used to achieve better performance.

Example Code

WXML

WXS

WXSS

```
<wxs src="./index.wxs" module="touch" />

<view class="container">
  <view class="videoBox"
        bind:touchstart="{{touch.touchstart}}"
        bind:touchmove="{{touch.touchmove}}"
        bind:touchend="{{touch.touchend}}">
    <video class="video-item" src="Video address 1" controls />
    <video class="video-item" src="Video address 2" controls />
    <video class="video-item" src="Video address 3" controls />
  </view>
</view>
```

```
// index.wxs
let startY = 0; // Starting Y coordinate
let isStart = false; // Whether the sliding has started
let moveDistance = 0; // Distance moved
let currentOffset = 0; // Save cumulative offset

// Touch start event
function touchstart(event, ins) {
  if (!isStart) {
    isStart = true;
    const touch = event.touches[0];
    startY = touch.pageY;
  }
}

// Touch move event
function touchmove(event, ins) {
  const { pageY } = event.touches[0];
  moveDistance = currentOffset + (pageY - startY);

  // Update element position
  ins.selectComponent('.videoBox').setStyle({
    transform: `translateY(${moveDistance}px)`
  });
}

// Touch end event
function touchend(event, ins) {
  isStart = false;
  currentOffset = moveDistance;
```

```
}

module.exports = {
  touchstart,
  touchmove,
  touchend
};
```

```
.container {
  width: 100vw;
  height: 100%;
  position: relative;
  overflow: hidden;
  top: 0;
  position: absolute;
  background: #000;
}

.videoBox {
  width: 100vw;
  height: 100%;
  position: absolute;
}

.video-item {
  width: 100vw;
  height: 100%;
  display: block;
}
```

3. Dynamic content update is limited

Delayed or flickering cover-view content updates (e.g. real-time text updates) over native components.

Solution:

Reduce high-frequency updates (e.g., control rendering frequency with an anti-jitter function), or switch to static content.

Example Code

WXML

JAVASCRIPT

```
<live-pusher>
  <cover-view class="danmu">{{danmuText}}</cover-view>
</live-pusher>
```

```
// utils/debounce.js
function debounce(fn, delay = 300, immediate = false) {
  let timer = null;
  return function (...args) {
    if (timer) clearTimeout(timer);

    if (immediate && !timer) {
      fn.apply(this, args); // Execute immediately once
    }

    timer = setTimeout(() => {
      if (!immediate) {
        fn.apply(this, args);
      }
      timer = null;
    }, delay);
  };
}
module.exports = debounce;



// pages/live.js
const debounce = require('../../utils/debounce');

Page({
  data: { danmuText: '' },

  onLoad() {
    // Create a debounced update function and bind the current Page's this
    this.debouncedUpdateDanmu = debounce(this.updateDanmu.bind(this), 300);
  },

  updateDanmu(text) {
    this.setData({ danmuText: text });
  }

  // Receive real-time WebSocket messages
  onSocketMessage(msg) {
    this.debouncedUpdateDanmu(msg.content); // High-frequency calls will be automat
  }
});
```

4. Bottom pop-up layer is limited

When you need to cover the full-screen native components (such as `video`, `live-player`) to realize the bottom pop-up layer, although it can be realized by cover-view, there will be the following limitations:
Interaction capability limitation：`cover-view` only supports nested `cover-view` and `cover-image`, so the popup layer can't contain complex interactions such as form submission.
Performance bottleneck：If a large number of `cover-views` or `cover-images` need to be inserted into the popup layer, it is easy to cause rendering lag.

Solution:

Dynamically adjust the height of the native component container so that there is no overlap between the popup layer and the native component in the layout, so that the popup layer can be realized by using a common component (e.g. `view`).
Example Code
WXML
JAVASCRIPT
WXSS

```
<view class="container">
  <live-pusher class="liver-pusher" url="{{pusherUrl}}" autopush="{{true}}"
mode="RTC">
    <cover-view class="action-button" catchtap="handleOpen">Open the pop-up
layer</cover-view>
    <cover-view class="action-button close" catchtap="handleClose">Close the
pop-up layer</cover-view>
  </live-pusher>
  <view class="comments-modal" wx:if="{{open}}">
    <input placeholder="This is an input"/>
  </view>
</view>
```

```
Page({
  data: {
    open: false,
  },
  handleOpen: function () {
    this.setData({
      open: true
    })
  },
  handleClose: function () {
    this.setData({
      open: false
    })
  },
```

```
})
```

```
.container {
  width: 100vw;
  height: 100vh;
  display: flex;
  flex-direction: column;
  background-color: #000;
}
.liver-pusher {
  width: 100%;
  flex: 1;
}
.action-button {
  position: absolute;
  height: 50px;
  line-height: 50px;
  text-align: center;
  background-color: white;
}

.close {
  left: 100px;
}

.comments-modal {
  width: 100%;
  background-color: #fff;
  height: 40vh;
}
```

View Container

Last updated：2025-05-26 10:49:39

scroll-view

Feature description:A scrollable view area. When using vertical scrolling, you need to give the `<scroll-view>` a fixed height, which can be set using WeiXin Style Sheets (WXSS).

Parameter and description:

Property	Type	Default value	Description
scroll-x	boolean	false	Allows horizontal scrolling.
scroll-y	boolean	false	Allows vertical scrolling.
upper-threshold	number/string	50	Distance from the top/left to trigger scrolltoupper.
lower-threshold	number/tring	50	Distance from the bottom/right to trigger scrolltolower.
scroll-top	number/string	-	Sets the vertical scroll bar position.
scroll-left	number/string	-	Sets the horizontal scroll bar position.
scroll-into-view	string	-	The value should be the id of a child element (id cannot start with a number). Scrolls to the element in the specified direction.
scroll-with-animation	boolean	false	Uses animation when setting the scroll bar position.
enable-back-to-top	boolean	false	On iOS, returns to the top when the status bar is tapped; on Android, returns to the top when the title bar is double-tapped. Only supports vertical scrolling.
refresher-enabled	boolean	false	Enables custom pull-down refresh.
refresher-threshold	number	45	Sets the custom pull-down refresh threshold.
refresher-default-style	string	black	Sets the default style for custom pull-down refresh. Options are `black | white |`

			`none` , `none` means no default style.
refresher-background	string	-	Sets the background color for the custom pull-down refresh area, default is transparent.
refresher-triggered	boolean	false	Sets the current pull-down refresh state. True means pull-down refresh has been triggered, false means it has not.
bounces	boolean	true	Controls the scroll-view boundary elasticity on iOS (effective when the enhanced property is enabled).
show-scrollbar	boolean	true	Controls the visibility of the scroll bar (effective when the enhanced property is enabled).
fast-deceleration	boolean	false	Controls the sliding deceleration rate, only effective on iOS (effective when the enhanced property is enabled).
binddragstart	eventhandle	-	Triggered when sliding starts (effective when the enhanced property is enabled). detail { scrollTop, scrollLeft }
binddragging	eventhandle	-	Triggered during sliding (effective when the enhanced property is enabled). detail { scrollTop, scrollLeft }
binddragend	eventhandle	-	Triggered when sliding ends (effective when the enhanced attribute is enabled). detail { scrollTop, scrollLeft, velocity }
bindscrolltoupper	eventhandle	-	Triggered when scrolled to the top/left
bindscrolltolower	eventhandle	-	Triggered when scrolled to the bottom/right
bindscroll	eventhandle	-	Triggered during scrolling. event.detail = {scrollLeft, scrollTop, scrollHeight, scrollWidth, deltaX, deltaY}
scroll-anchoring	boolean	false	Enables scroll anchoring, which adjusts the scroll position to compensate for the changes outside of the viewport. Only effective on iOS. For Android, refer to the Cascading Style Sheets (CSS) `overflow-anchor` property.
enhanced	boolean	false	Enables enhanced features for scroll-view.

			When enabled, you can operate the scroll-view through ScrollViewContext.
paging-enabled	boolean	false	Enables paging effect (effective when the enhanced property is enabled)

Notes:

The length of the scroll bar is estimated. If the height differences among direct child nodes are significant, the scroll bar length might be inaccurate.

Do not use textarea, map, canvas, or video components within a scroll-view.

The scroll-into-view has a higher priority than scroll-top.

Scrolling within a scroll-view will prevent the page from bouncing back, so onPullDownRefresh cannot be triggered within a scroll-view.

To use pull-down refresh, use the page's scrolling instead of scroll-view. This also allows returning to the top of the page by tapping the top status bar.

Example:

WXML

JAVASCRIPT

WXSS

```
<view class="container">
  <view class="page-body">
    <view class="page-section">
      <view class="page-section-title">
        <text>Vertical Scroll</text>
      </view>
      <view class="page-section-spacing">
        <scroll-view scroll-y="true" style="height: 300rpx;"
bindscrolltoupper="upper" bindscrolltolower="lower" bindscroll="scroll" scroll-
into-view="{{toView}}" scroll-top="{{scrollTop}}">
          <view id="demo1" class="scroll-view-item demo-text-1"></view>
          <view id="demo2"  class="scroll-view-item demo-text-2"></view>
          <view id="demo3" class="scroll-view-item demo-text-3"></view>
        </scroll-view>
      </view>
    </view>
    <view class="page-section">
      <view class="page-section-title">
        <text>Horizontal Scroll</text>
      </view>
      <view class="page-section-spacing">
        <scroll-view class="scroll-view_H" scroll-x="true" bindscroll="scroll"
style="width: 100%">
```

```
          <view id="demo1" class="scroll-view-item_H demo-text-1"></view>
          <view id="demo2"  class="scroll-view-item_H demo-text-2"></view>
          <view id="demo3" class="scroll-view-item_H demo-text-3"></view>
        </scroll-view>
      </view>
    </view>
  </view>
</view>
```

```
const order = ['demo1', 'demo2', 'demo3']

Page({
  onShareAppMessage() {
    return {
      title: 'scroll-view',
      path: 'page/component/pages/scroll-view/scroll-view'
    }
  },

  data: {
    toView: 'green'
  },

  upper(e) {
    console.log(e)
  },

  lower(e) {
    console.log(e)
  },

  scroll(e) {
    console.log(e)
  },

  scrollToTop() {
    this.setAction({
      scrollTop: 0
    })
  },

  tap() {
    for (let i = 0; i < order.length; ++i) {
      if (order[i] === this.data.toView) {
        this.setData({
          toView: order[i + 1],
```

```
            scrollTop: (i + 1) * 200
          })
          break
        }
      }
    },

    tapMove() {
      this.setData({
        scrollTop: this.data.scrollTop + 10
      })
    }
})
```

```
.page-section-spacing{
  margin-top: 60rpx;
}
.scroll-view_H{
  white-space: nowrap;
}
.scroll-view-item{
  height: 300rpx;
}
.scroll-view-item_H{
  display: inline-block;
  width: 100%;
  height: 300rpx;
}
```

swiper

Feature description:A swiper view container. `The change` event returns a `detail` object that includes a `source` field, indicating the cause of the change. Possible values are:

`autoplay:` The swiper changed due to automatic playback.

`touch:` The swiper changed due to user swiping.

An empty string indicates other reasons.

Parameter and description:

Property	Type	Default value	Description
indicator-dots	boolean	false	Specifies whether to show panel indicator dots.

indicator-color	color	rgba(0, 0, 0, .3)	Color of the indicator dots.
indicator-active-color	color	#000000	Color of the currently selected indicator dot.
autoplay	boolean	false	Specifies whether to enable automatic switching.
current	number	0	Index of the current slide
current-item-id	String	""	Item-id of the current slide, cannot be specified simultaneously with current.
interval	number	5,000	Time interval for automatic switching (in ms).
duration	number	500	Duration of the sliding animation (in ms).
circular	boolean	false	Specifies whether to enable circular swipe.
vertical	boolean	false	Specifies whether to enable vertical swipe
display-multiple-items	number	1	Number of items displayed simultaneously
previous-margin	string	"0px"	Margin before the current item, can be used to partially show the previous item. Accepts pixel and responsive pixel values.
easing-function	string	default	Specifies the easing animation type for swiper transitions. Valid values are: `default` : Default easing function. `linear` : Linear animation. `easeInCubic` : Ease-in animation. `easeOutCubic` : Ease out animation. `easeInOutCubic` : Ease in and out animation.
next-margin	string	"0px"	Margin after the current item, can be used to partially show the next item. Accepts pixel and responsive pixel values.
bindchange	eventhandle	-	Triggered when current changes. event.detail = {current: current, source: source}
bindtransition	eventhandle	-	Triggered when the position of a swiper-item changes. event.detail = {dx: dx, dy: dy}
bindanimationfinish	eventhandle	-	The animationfinish event is triggered when the animation ends. event.detail is the same as above

©2013-2025 Tencent Cloud International Pte. Ltd.

skip-hidden-item-layout	boolean	false	Whether to skip the layout of hidden items. Setting this to true can optimize swiping performance in complex scenarios but will lose the layout information of hidden items.

Notes:

Only swiper-item components can be placed inside a swiper. Placing other components may result in undefined behavior.

If you use setData to change the current value within the bindchange event callback, it may cause setData to be called repeatedly. Therefore, it is generally recommended to check the source field before changing the current value to determine if the change was caused by user touch.

Example:

WXML

JAVASCRIPT

```
<view class="container">
  <view class="page-body">
    <view class="page-section page-section-spacing swiper">
      <swiper indicator-dots="{{indicatorDots}}"
        autoplay="{{autoplay}}" interval="{{interval}}" duration="
{{duration}}">
        <block wx:for="{{background}}" wx:key="*this">
          <swiper-item>
            <view class="swiper-item {{item}}"></view>
          </swiper-item>
        </block>
      </swiper>
    </view>
    <view class="page-section" style="margin-top: 40rpx;margin-bottom: 0;">
      <view class="weui-cells weui-cells_after-title">
        <view class="weui-cell weui-cell_switch">
          <view class="weui-cell__bd">Indication Point</view>
          <view class="weui-cell__ft">
            <switch checked="{{indicatorDots}}"
bindchange="changeIndicatorDots" />
          </view>
        </view>
        <view class="weui-cell weui-cell_switch">
          <view class="weui-cell__bd">Autoplay</view>
          <view class="weui-cell__ft">
            <switch checked="{{autoplay}}" bindchange="changeAutoplay" />
          </view>
        </view>
      </view>
```

```
    </view>

    <view class="page-section page-section-spacing">
      <view class="page-section-title">
        <text>Slide switching duration (ms)</text>
        <text class="info">{{duration}}</text>
      </view>
      <slider bindchange="durationChange" value="{{duration}}" min="500"
max="2000"/>
      <view class="page-section-title">
        <text>Automatic playback interval duration (ms)</text>
        <text class="info">{{interval}}</text>
      </view>
      <slider bindchange="intervalChange" value="{{interval}}" min="2000"
max="10000"/>
    </view>
  </view>
</view>
```

```
Page({
  onShareAppMessage() {
    return {
      title: 'swiper',
      path: 'page/component/pages/swiper/swiper'
    }
  },

  data: {
    background: ['demo-text-1', 'demo-text-2', 'demo-text-3'],
    indicatorDots: true,
    vertical: false,
    autoplay: false,
    interval: 2000,
    duration: 500
  },

  changeIndicatorDots() {
    this.setData({
      indicatorDots: !this.data.indicatorDots
    })
  },

  changeAutoplay() {
    this.setData({
      autoplay: !this.data.autoplay
    })
```

```
  },

  intervalChange(e) {
    this.setData({
      interval: e.detail.value
    })
  },

  durationChange(e) {
    this.setData({
      duration: e.detail.value
    })
  }
})
```

swiper-item

Feature description: Can only be placed in a swiper component. Its width and height are automatically set to 100%.

Parameter and description:

Property	Type	Default value	Description
item-id	string	""	Identifier of the swiper-item.

view

Feature description:A view container.

Parameter and description:

Property	Type	Default value	Description
hover-class	string	-	Specifies the style class when tapped. No tap effect if `hover-class="none"`.
hover-stop-propagation	boolean	false	Specifies whether to prevent ancestor nodes from showing the tap effect.
hover-start-time	number	50	Time in milliseconds before the tap effect appears after pressing.
hover-stay-time	number	400	Time in milliseconds the tap effect remains after the finger is released.

Notes:

If you need to use a scroll view, please use scroll-view.

Example:

WXML

```
<view class="container">
  <view class="page-body">
    <view class="page-section">
      <view class="page-section-title">
        <text>flex-direction: row\\nHorizontal layout</text>
      </view>
      <view class="page-section-spacing">
        <view class="flex-wrp" style="flex-direction:row;">
          <view class="flex-item demo-text-1"></view>
          <view class="flex-item demo-text-2"></view>
          <view class="flex-item demo-text-3"></view>
        </view>
      </view>
    </view>
    <view class="page-section">
      <view class="page-section-title">
        <text>flex-direction: column\\nVertical layout</text>
      </view>
      <view class="flex-wrp" style="flex-direction:column;">
        <view class="flex-item flex-item-V demo-text-1"></view>
        <view class="flex-item flex-item-V demo-text-2"></view>
        <view class="flex-item flex-item-V demo-text-3"></view>
      </view>
    </view>
  </view>
</view>
```

movable-area

Feature description: Defines the movable area for movable-view .

Parameter and description:

Property	Type	Default value	Description
scale-area	boolean	false	When the movable-view is set to support pinch-to-zoom, setting this value will make the zoom gesture effective for the entire movable-area.

Notes:

The movable-area must have width and height attributes set. If not set, the default value is 10px.

When movable-view is smaller than movable-area, the moving range of movable-view is within movable-area.

When movable-view is larger than movable-area, the movement range of movable-view must encompass the movable-area (considering the x-axis and y-axis separately).

Example:

WXML

JAVASCRIPT

```
<view class="container">

  <view class="page-body">
    <view class="page-section">
      <view class="page-section-title first">The movable-view area is smaller
than the movable-area</view>
      <movable-area>
        <movable-view x="{{x}}" y="{{y}}" direction="all">text</movable-view>
      </movable-area>
    </view>
    <view class="btn-area">
      <button bindtap="tap" class="page-body-button" type="primary">Tap to move
to (30px, 30px)</button>
    </view>

    <view class="page-section">
      <view class="page-section-title">The movable-view area is larger than the
movable-area</view>
      <movable-area>
        <movable-view class="max" direction="all">text</movable-view>
      </movable-area>
    </view>

    <view class="page-section">
      <view class="page-section-title">Only horizontal movement allowed</view>
      <movable-area>
        <movable-view direction="horizontal">text</movable-view>
      </movable-area>
    </view>

    <view class="page-section">
      <view class="page-section-title">Only vertical movement allowed</view>
      <movable-area>
        <movable-view direction="vertical">text</movable-view>
      </movable-area>
    </view>
```

```
    <view class="page-section">
      <view class="page-section-title">Allow out of bounds</view>
      <movable-area>
        <movable-view direction="all" out-of-bounds>text</movable-view>
      </movable-area>
    </view>

    <view class="page-section">
      <view class="page-section-title">With inertia</view>
      <movable-area>
        <movable-view direction="all" inertia>text</movable-view>
      </movable-area>
    </view>

    <view class="page-section">
      <view class="page-section-title">Scalable</view>
      <movable-area scale-area>
        <movable-view direction="all" bindchange="onChange" bindscale="onScale"
scale scale-min="0.5" scale-max="4" scale-value="{{scale}}">text</movable-view>
      </movable-area>
    </view>

    <view class="btn-area">
      <button bindtap="tap2" class="page-body-button" type="primary">Tap to
zoom in 3 times</button>
    </view>
  </view>

</view>
```

```
Page({
  onShareAppMessage() {
    return {
      title: 'movable-view',
      path: 'page/component/pages/movable-view/movable-view'
    }
  },

  data: {
    x: 0,
    y: 0,
    scale: 2,
  },

  tap() {
```

```
    this.setData({
      x: 30,
      y: 30
    })
  },

  tap2() {
    this.setData({
      scale: 3
    })
  },

  onChange(e) {
    console.log(e.detail)
  },

  onScale(e) {
    console.log(e.detail)
  }
})
```

movable-view

Feature description:A movable view container that can be dragged and slid within a page.movable-view must be a direct child of the movable-area component to be movable.

Parameter and description:

Property	Type	Default value	Description
direction	string	none	The direction in which the movable-view can move. Possible values are all, vertical, horizontal, and none.
inertia	boolean	false	Whether the movable-view has inertia.
out-of-bounds	boolean	false	Whether the movable-view can move beyond the movable area.
x	number/string	0	Defines the x-axis offset. If the value is out of the movable range, it will automatically move to the nearest valid position. Changing this value triggers an animation.
y	number/string	0	Defines the y-axis offset. If the value is out of the movable range, it will automatically move to the

			nearest valid position. Changing this value triggers an animation.
damping	number	20	The damping coefficient, which controls the animation when x or y changes and the bounce-back animation when out of bounds. The larger the value, the faster the movement.
friction	number	2	The friction coefficient, which controls the animation of inertial sliding. The larger the value, the greater the friction and the quicker the stop. Must be greater than 0, otherwise it defaults to the set value.
disabled	boolean	false	Whether the movable-view is disabled.
scale	boolean	false	Whether the movable-view supports pinch-to-zoom. By default, the zoom gesture is effective within the movable-view.
scale-min	number	0.5	Defines the minimum zoom scale.
scale-max	number	10	Defines the maximum zoom scale.
scale-value	number	1	Defines the zoom scale, with a range of 0.5 to 10.
animation	boolean	true	Whether to use animation when changing the zoom scale.
bindchange	eventhandle	-	Events triggered during dragging, event.detail = {x: x, y: y, source: source}, where source indicates the reason for the movement. Possible values are: `touch` : Dragging. `touch-out-of-bounds` : Out of bounds. `out-of-bounds` : Rebound back after out of bounds. `friction` : Inertia. `Empty string` : setData.
bindscale	eventhandle	-	Event triggered during scaling. event.detail = {x: x, y: y, scale: scale}

In addition to basic events, movable-view provides two special events.

Event name	Type	Trigger condition
htouchmove	eventhandle	Triggered when the initial finger touch moves horizontally. Catching this event also catches the touchmove event.

vtouchmove	eventhandle	Triggered when the initial finger touch moves vertically. Catching this event also catches the touchmove event.

Notes:

The movable-view must have width and height attributes set. If not set, the default value is 10px.

movable-view is positioned absolutely by default, with top and left properties set to 0px.

cover-image

Feature description: An image view that overlays native components, similar to cover-view. It can be nested within a cover-view .

Parameter and description:

Property	Type	Default value	Description
src	string	-	The path to the image. Supports temporary paths and network addresses. `SVG、GIF` and `BASE64` formats are not supported.
bindload	eventhandle	-	Event triggered when the image loads successfully.
binderror	eventhandle	-	Event triggered when the image fails to load.

cover-view

Feature description:A text view that overlays native components. It can cover components such as map,video,Canvas,camera,live-player and live-pusher. It supports nesting cover-view and cover-image, and you can use button within a cover-view.

Parameter and description:

Property	Type	Default value	Description
scroll-top	number / string	-	Sets the top scroll offset. This only takes effect when overflow-y: scroll is set, making it a scrollable element.

Notes:

The event model follows the bubbling model but will not bubble up to native components.

©2013-2025 Tencent Cloud International Pte. Ltd.

It is recommended to wrap all text in cover-view tags to avoid layout issues.

Only basic positioning, layout, and text styles are supported. It does not support setting individual borders, background images, shadows, or overflow: visible.

It is recommended that child nodes do not overflow their parent nodes.

You can't use z-index to control the stacking order.

The default styles include: white-space: nowrap; line-height: 1.2; display: block.

When nesting cover-view within custom components, the slot of the custom component and its parent nodes do not support visibility control via wx:if, as this will cause the cover-view to not display.

Example:

WXML

JAVASCRIPT

WXSS

```
<view class="container">
  <view class="page-body">
    <view class="page-section page-section-gap">
      <map
        style="width: 100%; height: 300px;"
        latitude="{{latitude}}"
        longitude="{{longitude}}"
      >
        <cover-view class="cover-view">
          <cover-view class="container">
            <cover-view class="flex-wrp" style="flex-direction:row;">
              <cover-view class="flex-item demo-text-1"></cover-view>
              <cover-view class="flex-item demo-text-2"></cover-view>
              <cover-view class="flex-item demo-text-3"></cover-view>
            </cover-view>
          </cover-view>
        </cover-view>
      </map>
    </view>
  </view>
</view>
```

```
Page({
  onShareAppMessage() {
    return {
      title: 'cover-view',
      path: 'page/component/pages/cover-view/cover-view'
    }
  },

  data: {
```

```
    latitude: 23.099994,
    longitude: 113.324520,
  }
})
```

```
.cover-view {
  position: absolute;
  top: calc(50% - 150rpx);
  left: calc(50% - 300rpx);
  /* opacity: .7; */
}

.flex-wrp{
  display:flex;
}

.flex-item{
  width: 200rpx;
  height: 300rpx;
  font-size: 26rpx;
}

.demo-text-1 {
  background: rgba(26, 173, 25, 0.7);
}

.demo-text-2 {
  background: rgba(39, 130, 215, 0.7);
}

.demo-text-3 {
  background: rgba(255, 255, 255, 0.7);
}
```

Fundamentals

Last updated：2025-05-26 10:51:38

icon

Feature description: Icon component.

Parameter and description:

Property	Type	Default value	Description
type	string	-	The type of icon. Valid values are: success, success_no_circle, info, warn, waiting, cancel, download, search, clear.
size	umber/string	23	The size of the icon. The default unit is pixel.
color	string	-	The color of the icon, following the same rules as Cascading Style Sheets (CSS) color.

Example:

WXML

JAVASCRIPT

```
<view class="container">
  <view class="icon-box">
    <icon class="icon-box-img" type="success" size="93"></icon>
    <view class="icon-box-ctn">
      <view class="icon-box-title">Success</view>
      <view class="icon-box-desc">Used to indicate a successful operation.
</view>
    </view>
  </view>
  <view class="icon-box">
    <icon class="icon-box-img" type="info" size="93"></icon>
    <view class="icon-box-ctn">
      <view class="icon-box-title">Info</view>
      <view class="icon-box-desc">Used for informational prompts and to
intercept operations lacking necessary conditions, prompting the user for
required information.</view>
    </view>
  </view>
  <view class="icon-box">
```

```
    <icon class="icon-box-img" type="warn" size="93" color="#C9C9C9"></icon>
    <view class="icon-box-ctn">
      <view class="icon-box-title">Warning</view>
      <view class="icon-box-desc">Indicates potential consequences of an
operation or negative outcomes due to system issues.</view>
    </view>
  </view>
  <view class="icon-box">
    <icon class="icon-box-img" type="warn" size="93"></icon>
    <view class="icon-box-ctn">
      <view class="icon-box-title">Strong warning</view>
      <view class="icon-box-desc">Indicates negative outcomes from user actions
or irreversible serious consequences.</view>
    </view>
  </view>
  <view class="icon-box">
    <icon class="icon-box-img" type="waiting" size="93"></icon>
    <view class="icon-box-ctn">
      <view class="icon-box-title">Waiting</view>
      <view class="icon-box-desc">Indicates that the user needs to wait for the
result.</view>
    </view>
  </view>
  <view class="icon-box">
    <view class="icon-small-wrp">
      <icon class="icon-small" type="success_no_circle" size="23"></icon>
    </view>
    <view class="icon-box-ctn">
      <view class="icon-box-title">Multi-select icon_Selected</view>
      <view class="icon-box-desc">Used in the multi-select control to indicate
that the item has been selected.</view>
    </view>
  </view>
  <view class="icon-box">
    <view class="icon-small-wrp">
      <icon class="icon-small" type="circle" size="23"></icon>
    </view>
    <view class="icon-box-ctn">
      <view class="icon-box-title">Multi-select icon_Not selected</view>
      <view class="icon-box-desc">Used in the multi-select control to indicate
that the item is selectable but has not been selected.</view>
    </view>
  </view>
  <view class="icon-box">
    <view class="icon-small-wrp">
      <icon class="icon-small" type="warn" size="23"></icon>
    </view>
```

```
    <view class="icon-box-ctn">
      <view class="icon-box-title">Error</view>
      <view class="icon-box-desc">Indicates that an error occurs in the form.
</view>
    </view>
  </view>
  <view class="icon-box">
    <view class="icon-small-wrp">
      <icon class="icon-small" type="success" size="23"></icon>
    </view>
    <view class="icon-box-ctn">
      <view class="icon-box-title">Single-select icon_Selected</view>
      <view class="icon-box-desc">Used in the single-select control to indicate
that the item has been selected.</view>
    </view>
  </view>
  <view class="icon-box">
    <view class="icon-small-wrp">
      <icon class="icon-small" type="download" size="23"></icon>
    </view>
    <view class="icon-box-ctn">
      <view class="icon-box-title">Download</view>
      <view class="icon-box-desc">Indicates that the item is downloadable.
</view>
    </view>
  </view>
  <view class="icon-box">
    <view class="icon-small-wrp">
      <icon class="icon-small" type="info_circle" size="23"></icon>
    </view>
    <view class="icon-box-ctn">
      <view class="icon-box-title">Info</view>
      <view class="icon-box-desc">Indicates that there is an information prompt
in the form.</view>
    </view>
  </view>
  <view class="icon-box">
    <view class="icon-small-wrp">
      <icon class="icon-small" type="cancel" size="23"></icon>
    </view>
    <view class="icon-box-ctn">
      <view class="icon-box-title">Stop or close</view>
      <view class="icon-box-desc">Indicates close or stop in the form.</view>
    </view>
  </view>
  <view class="icon-box">
    <view class="icon-small-wrp">
```

```
      <icon class="icon-small" type="search" size="14"></icon>
    </view>
    <view class="icon-box-ctn">
      <view class="icon-box-title">Search</view>
      <view class="icon-box-desc">Used in the search control to indicate that
search can be implemented.</view>
    </view>
  </view>
</view>
```

```
Page({
  data: {
    iconSize: [20, 30, 40, 50, 60, 70],
    iconColor: [
      'red', 'orange', 'yellow', 'green', 'rgb(0,255,255)', 'blue', 'purple'
    ],
    iconType: [
      'success', 'success_no_circle', 'info', 'warn', 'waiting', 'cancel', 'downloa
    ]
  }
})
```

progress

Feature description: Progress bar.

Parameter and description:

Property	Type	Default value	Description
percent	float	-	Percentage (0-100).
show-info	boolean	false	Displays the percentage on the right side of the progress bar.
border-radius	number / string	0	Border radius in pixel.
font-size	number / string	16	Font size of the percentage on the right side in pixel.
stroke-width	number / string	6	Width of the progress bar line.

color	color	#09BB07	Progress bar color (use activeColor)
activeColor	color	#09BB07	Color of the selected progress bar.
backgroundColor	color	#EBEBEB	Color of the unselected progress bar.
active	boolean	false	Animation of the progress bar moving from left to right.
active-mode	string	backwards	backwards: Animation starts from the beginning. forwards: Animation continues from the last end point.
bindactiveend	eventhandle	-	Event triggered when the animation completes.

Example:

WXML

```
<view class="progress-box">
  <progress percent="20" show-info stroke-width="3"/>
</view>

<view class="progress-box">
  <progress percent="40" active stroke-width="3" />
  <icon class="progress-cancel" type="cancel"></icon>
</view>

<view class="progress-box">
  <progress percent="60" active stroke-width="3" />
</view>

<view class="progress-box">
  <progress percent="80" color="#10AEFF" active stroke-width="3" />
</view>
```

text

Feature description: Text.

Parameter and description:

Property	Type	Default value	Description
selectable	boolean	false	Whether the text is selectable.
space	string	-	Displays continuous spaces: ensp: Half-width space of a Chinese character

			emsp: Full-width space of a Chinese character nbsp: Space size based on font settings.
decode	boolean	false	Whether to decode the character entities in the text.

Notes:

Decode can be parsed with `< > & '    ` .

The standard of whitespace is not consistent across operating systems.

The text component only supports nesting of text elements.

Nodes other than text cannot be pressed and held to select.

WXML

JAVASCRIPT

```
<view class="container">
  <view class="page-body">
    <view class="page-section page-section-spacing">
      <view class="text-box" scroll-y="true" scroll-top="{{scrollTop}}">
        <text>{{text}}</text>
      </view>
      <button disabled="{{!canAdd}}" bindtap="add">add line</button>
      <button disabled="{{!canRemove}}" bindtap="remove">remove line</button>
    </view>
  </view>
</view>
```

```
const texts = [
    'lin1: text1',
    'lin2: text2',
    'lin3: text3',
    'lin4: text4',
    'lin5: text5',
    'lin6: text6',
    'lin7: text7',
    'lin8: text8',
    'lin9: text9',
    'lin10: text10',
    'lin11: text11',
    'lin12: text12',
    '......'
  ]

Page({
  onShareAppMessage() {
    return {
```

```
      title: 'text',
      path: 'page/component/pages/text/text'
    }
  },

  data: {
    text: '',
    canAdd: true,
    canRemove: false
  },
  extraLine: [],

  add() {
    this.extraLine.push(texts[this.extraLine.length % 12])
    this.setData({
      text: this.extraLine.join('\\n'),
      canAdd: this.extraLine.length < 12,
      canRemove: this.extraLine.length > 0
    })
    setTimeout(() => {
      this.setData({
        scrollTop: 99999
      })
    }, 0)
  },
  remove() {
    if (this.extraLine.length > 0) {
      this.extraLine.pop()
      this.setData({
        text: this.extraLine.join('\\n'),
        canAdd: this.extraLine.length < 12,
        canRemove: this.extraLine.length > 0,
      })
    }
    setTimeout(() => {
      this.setData({
        scrollTop: 99999
      })
    }, 0)
  }
})
```

rich-text

Feature description: Rich text.

Parameter and description:

Property	Type	Default value	Description
nodes	array/string	[]	List of nodes/HTML string.
space	string	-	Displays continuous spaces: ensp: Half-width space of a Chinese character emsp: Full-width space of a Chinese character nbsp: Space size based on font settings.

Supports default events,

including: `tap` , `touchstart` , `touchmove` , `touchcancel` , `touchend` and `longtap` .

nodes

The nodes property is recommended to be used as an Array type. Using a String type will result in a performance decrease since the component converts it to an Array. Two types of nodes are supported, distinguished by the type: Element nodes and text nodes. By default, it is an element node, which is the displayed HTML nodes within the rich text area.

Element node: type = node

Property	Description	Type	Required	Note
name	Tag name	string	True	Supports a subset of trusted HTML nodes.
attrs	Property	object	False	Supports a subset of trusted properties, following PascalCase naming convention.
children	List of child nodes	array	False	The structure is the same as the nodes.

Text node: type=text

Property	Description	Type	Required	Note
text	Text	string	True	Supports entities.

Trusted HTML nodes and properties

Globally supports the class and style properties,**but does not support the id property.**

Node	Property	Node	Property
a	-	img	alt,src,height,width
abbr	-	ins	-

b	-	label	-
br	-	legend	-
code	-	li	-
col	span,width	ol	start,type
colgroup	span,width	p	-
dd	-	q	-
del	-	span	-
div	-	strong	-
dl	-	sub	-
dt	-	sup	-
em	-	table	width
fieldset	-	tbody	-
h1	-	td	colspan,height,rowspan,width
h2	-	tfoot	-
h3	-	th	colspan,height,rowspan,width
h4	-	thead	-
h5	-	tr	-
h6	-	ul	-
hr	-	-	-

Notes:

Using a String type for nodes is not recommended due to performance degradation.

All node events within the rich-text component will be blocked to ensure that users cannot tap, touch, or trigger any events.

The attrs property does not support id, but it does support class.

The name property is case-insensitive.

If an untrusted HTML node is used, the nodes and all their children will be removed.

The img tag only supports online images.

©2013-2025 Tencent Cloud International Pte. Ltd.

If the rich-text component is used within a custom component, only the custom component's WeiXin Style Sheets (WXSS) styles will apply to classes within rich-text.

Example

WXML

JAVASCRIPT

```
<view class="container">
  <view class="page-body">
    <view class="page-section">
      <view class="page-section-title">Render via HTML String</view>
      <view class="page-content">
        <scroll-view scroll-y>{{htmlSnip}}</scroll-view>
        <button style="margin: 30rpx 0" type="primary"
bindtap="renderHtml">Render HTML</button>
        <block wx:if="{{renderedByHtml}}">
          <rich-text nodes="{{htmlSnip}}"></rich-text>
        </block>
      </view>
    </view>

    <view class="page-section">
      <view class="page-section-title">Render via nodes</view>
      <view class="page-content">
        <scroll-view scroll-y>{{nodeSnip}}</scroll-view>
        <button style="margin: 30rpx 0" type="primary"
bindtap="renderNode">Render nodes</button>
        <block wx:if="{{renderedByNode}}">
          <rich-text nodes="{{nodes}}"></rich-text>
        </block>
      </view>
    </view>
  </view>
</view>
```

```
const htmlSnip =
`<div class="div_class">
  <h1>Title</h1>
  <p class="p">
    Life is <i>like</i> a box of
    <b> chocolates</b>.
  </p>
</div>
`

const nodeSnip =
```

```
`Page({
  data: {
    nodes: [{
      name: 'div',
      attrs: {
        class: 'div_class',
        style: 'line-height: 60px; color: red;'
      },
      children: [{
        type: 'text',
        text: 'You never know what you're gonna get.'
      }]
    }]
  }
})
`

Page({
  onShareAppMessage() {
    return {
      title: 'rich-text',
      path: 'page/component/pages/rich-text/rich-text'
    }
  },

  data: {
    htmlSnip,
    nodeSnip,
    renderedByHtml: false,
    renderedByNode: false,
    nodes: [{
      name: 'div',
      attrs: {
        class: 'div_class',
        style: 'line-height: 60px; color: #1AAD19;'
      },
      children: [{
        type: 'text',
        text: 'You never know what you\\'re gonna get.'
      }]
    }]
  },
  renderHtml() {
    this.setData({
      renderedByHtml: true
    })
  },
```

```
  renderNode() {
    this.setData({
      renderedByNode: true
    })
  },
  enterCode(e) {
    console.log(e.detail.value)
    this.setData({
      htmlSnip: e.detail.value
    })
  }
})
```

Map Component

Last updated：2024-12-19 16:37:00

map

Feature description: Map. This component is a Native Component, and there are some restrictions for using it.

Related API: wx.createMapContext.

Support:

System map: Supported only on iOS.

Google Maps: Supported on Android and in IDE.

Huawei Maps: Supported only on Android.

Tencent Maps: Supported on iOS, Android, and in IDE.

Parameter and description:

Property	Type	Default value	Required	Description	Supported on iOS	Supported on Android
type	string	tencent	False	Map type, either Tencent or Google.	False	False
longitude	number	-	True	Center longitude.	True	True
latitude	number	-	True	Center latitude.	True	True
scale	number	16	False	Zoom level. Value range: 3-20.	True	True
min-scale	number	3	False	Minimum zoom level.		True
max-scale	number	20	False	Maximum zoom level.		True
markers	array.<marker>	-	False	Markers.	True	True
polyline	array.<polyline>	-	False	Routes.	True	True

circles	array.<circle>	-	False	Circles.	True	True
controls	Array.<control>	-	False	Controls (to be deprecated, it is recommended to use cover-view instead).	False	False
include-points	array.<point>	-	False	Zooms to include all given coordinates	True	True
show-location	boolean	false	False	Shows current location with direction.	True	True
polygons	array.<polygon>	-	False	Polygons.	True	True
subkey	string	-	False	The key for the custom maps.	False	False
layer-style	number	1	False	The style for the custom maps.	False	False
rotate	number	0	False	Rotation angle. Value range: 0-360. It is the angle between true north on the map and the device's y-axis.	False	True
skew	number	0	False	Tilt angle. Valid range 0-40. It is the tilt	System map: False	True

				angle around the Z-axis.	Tencent Maps: True Baidu Maps: True Amap: True		
enable-3D	boolean	false	False	Shows the 3D building blocks.	System map: False Tencent Maps: True Baidu Maps: False Amap: False	Google Maps: True Huawei Maps: True Tencent Maps: True Baidu Maps: True Amap: False	
show-compass	boolean	false	False	Shows the compass.	True	True	
show-scale	boolean	false	False	Show the scale.	True	Google Maps: False Huawei Maps: False Tencent Maps: True Baidu Maps: True Amap: True	
enable-overlooking	boolean	false	False	Enables the top view.	System map: False	True	

					Tencent Maps: True Baidu Maps: True Amap: True	
enable-zoom	boolean	true	False	Whether to support zoom.	True	True
enable-scroll	boolean	true	False	Whether to support scrolling.	True	True
enable-rotate	boolean	-	False	Whether to support rotation.	True	True
enable-satellite	boolean	false	False	Whether to enable the satellite map.	True	Google Maps: True Huawei Maps: False Tencent Maps: True Baidu Maps: True Amap: True
enable-traffic	boolean	false	False	Whether to enable real-time traffic information.	True	True
enable-poi	boolean	true	False	Whether to display the point of interest (POI).	System map: False Tencent Maps: True	Google Maps: True Huawei Maps: False

					Baidu Maps: False Amap: False	Tencent Maps: True Baidu Maps: True Amap: True
enable-building	boolean	true	False	Whether to display the buildings.	System map: False Tencent Maps: True Baidu Maps: True Amap: True	Google Maps: True Huawei Maps: False Tencent Maps: True Baidu Maps: False Amap: True
setting	object	-	False	Configuration options.	False	False
bindtap	eventhandle	-	False	Triggered when the user taps the map, which returns the latitude and longitude information.	True	True
bindmarkertap	eventhandle	-	False	Triggered when the user taps the marker, e.detail = {markerId}.	True	True
bindlabeltap	eventhandle	-	False	Triggered when the user taps the label,	True	False

				e.detail = {markerId}.			
bindcontroltap	eventhandle	-	False	Triggered when the user taps the control, e.detail = {controlId}.	False	False	
bindcallouttap	eventhandle	-	False	Triggered when the user taps the marker's callout bubble, e.detail = {markerId}.	False	False	
bindupdated	eventhandle	-	False	Triggered when the map rendering is completed.	False	True	
bindregionchange	eventhandle	-	False	Triggered when the view changes.	True	True	
bindpoitap	eventhandle	-	False	Triggered when the user taps the POI on the map, e.detail = {name, longitude, latitude}.	False	True	
bindanchorpointtap	eventhandle	-	False	Triggered when the user taps the location marker, e.detail = {longitude, latitude}.	False	False	

©2013-2025 Tencent Cloud International Pte. Ltd.

Notes:

Color values in the map, such as color, borderColor, and bgColor, must be represented using a 6-digit (or 8-digit) hexadecimal format. When using 8 digits, the last two digits represent the alpha value, e.g., #000000AA.
Latitude and longitude are required fields for the map component. If not specified, the default values will be set to the coordinates of Beijing.
The map component uses the GCJ-02 coordinate system. When calling the wx.getLocation, you need to specify the type as gcj02.

Returned values of regionchange

When the view changes, the regionchange event is triggered twice, with the returned type values being begin and end.
Begin stage: Returns causedBy with valid values: gesture (triggered by a gesture) and update (triggered by an API update).
End stage: Returns causedBy with valid values: drag (caused by dragging), scale (caused by scaling) and update (caused by calling an update API).

```
e = {causedBy, type, detail: {rotate, skew, scale, centerLocation, region}}
```

setting

A setting object is provided to uniformly set map configurations. For certain animation properties like rotate and skew, they cannot take effect simultaneously if set separately using setData. Instead, they need to be modified together through the setting object.

```
// Default values
const setting = {
  rotate: 0,
  showLocation: false,
  subKey: '',
  layerStyle: 1,
  enableZoom: true,
  enableScroll: true,
  enableRotate: false,
  showCompass: false,
  enableSatellite: false,
}

this.setData({
  // Changes only affect the properties involved in the setting object; other prope
  setting: {
    enableZoom: false,
    enableScroll: false,
  }
})
```

marker

Markers, which are used to display the specific locations on the map.

Property	Description	Type	Required	Note	Supported on iOS	Supported on Android
id	Marker ID.	number	False	This ID will be returned in the callback when the marker is tapped. It is recommended to set an ID for each marker to ensure better performance when updating the markers.	True	True
clusterId	Cluster ID.	number	False	Used for custom point clustering effects.	False	False
joinCluster	Whether to participate in clustering.	boolean	False	Default is not to participate in clustering.	False	Google Maps: True Huawei Maps: True Tencent Maps: True Baidu Maps: False Amap: False
latitude	The latitude.	number	True	Floating point number. Value	True	True

				range: -90 - 90.		
longitude	The longitude.	number	True	Floating point number. Value range: -90 - 90.	True	True
title	Marker name.	string	False	Displayed when tapped, ignored if callout exists.	True	True
zIndex	Displays level.	number	False	-	False	True
iconPath	Icon path.	string	True	Path to the image in the project directory, supports network path, local path, and code package path.	True	True
rotate	Rotation angle.	number	False	Clockwise rotation angle, range 0-360, default is 0.	False	True
alpha	Marker transparency.	number	False	Default value: 1, which means non-transparency. Value range: 0-1.	False	True
width	Width of the marker icon.	number/string	False	It defaults to the actual width of the image.	False	True
height	Height of the marker icon.	number/string	False	It defaults to the actual	False	True

				height of the image.		
callout	Callout bubble window above the marker.	object	False	See the table below for supported properties, and line breaks can be recognized.	True	True
customCallout	Custom callout bubble window.	object	False	See the table below for supported properties.	False	False
label	Adds labels next to the markers.	object	False	See the table below for supported properties, and line breaks can be recognized.	False	False
anchor	The anchor point of the latitude and longitude on the the marker icon, default is the bottom center.	object	False	{x, y}, where x represents horizontal (0-1) and y represents vertical (0-1). {x: .5, y: 1} means the bottom center.	False	True

Callout bubble above the marker

Property	Description	Type	Supported on iOS	Supported on Android	Supported in IDE
content	Text.	string	True	True	True
color	Text color.	string	False	True	True
fontSize	Font size.	number	False	True	True
borderRadius	Border radius.	number	False	True	True

borderWidth	Width of the border.	number	False	True	True
borderColor	Border color.	string	False	True	True
bgColor	Background color.	string	False	True	True
padding	Padding around the text.	number	False	True	True
display	'BYCLICK' (show when tapped), 'ALWAYS' (always show)	string	False	False	True
textAlign	Text alignment. Valid values: left, right, center.	string	False	True	False
anchorX	Horizontal offset, positive values move right.	number	False	Google Maps: True Huawei Maps: True Tencent Maps: True Baidu Maps: False Amap: True	False
anchorY	Vertical offset, positive values move down.	number	False	Google Maps: True Huawei Maps: True Tencent Maps: True Baidu Maps: False Amap: True	False

Custom callout properties for markers

`When customCallout` is present, the `callout` and `title` properties will be ignored. Custom callout bubbles use `cover-view` for higher flexibility.

Property	Description	Type	Supported on iOS	Supported on Android	Supported in IDE
display	'BYCLICK' (show when	string	False	False	False

	tapped), 'ALWAYS' (always show)				
anchorX	Horizontal offset, positive values move right.	number	False	False	False
anchorY	Vertical offset, positive values move down.	number	False	False	False

To use a custom callout, add a slot node named callout under the `map component`. Inside this slot, use `cover-view` with the `marker-id` property to bind it to the `marker`. When the `marker` is created, the content displayed in this `cover-view` will appear as the `callout` above the marker.

```
<map>
  <cover-view slot="callout">
    <cover-view marker-id="1"></cover-view>
    <cover-view marker-id="2"></cover-view>
  </cover-view>
</map>
```

Label properties for markers

Property	Description	Type	Supported on iOS	Supported on Android	Supported in IDE
content	Text.	string	False	False	True
color	Text color.	string	False	False	True
fontSize	Font size.	number	False	False	True
anchorX	The coordinates of the label, with the origin being the latitude and longitude of the marker.	number	False	False	True
anchorY	The coordinates of the label, with the origin being the latitude and longitude of the marker.	number	False	False	True
borderWidth	Width of the border.	number	False	False	True
borderColor	Border color.	string	False	False	True
borderRadius	Border radius.	number	False	False	True

©2013-2025 Tencent Cloud International Pte. Ltd.

bgColor	Background color.	string	False	False	True
padding	Padding around the text.	number	False	False	True
textAlign	Text alignment. Valid values: left, right, center.	string	False	False	True

Marker clustering

Feature description: When there are too many markers on the map, it can lead to marker overlap, incomplete display, and decreased overall performance. To address these issues, marker clustering feature is introduced.

System map (iOS only)

Directions:

Use the MapContext.addMarkers to specify the markers for clustering.

(Optional) `When MapContext.on('markerClusterCreate', callback)` is triggered, update the style of the cluster by using MapContext.addMarkers .

Use the MapContext.removeMarkers to remove the markers involved in the clustering.

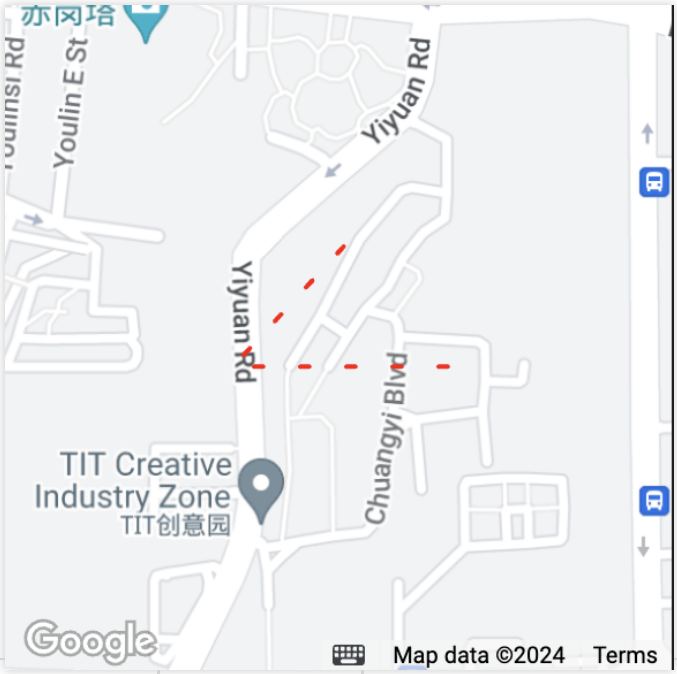
Google Maps (Android only)

Specify the joinCluster when adding markers, and the markers that meet the conditions will be automatically clustered.

polyline

Feature description:A polyline connects a series of coordinates, drawing a line from the first point in the array to the last. When drawing a rainbow line, you need to specify different colors for each segment. For example, if 5 points are contained, colorList should have 4 color values. If colorList is shorter than points.length - 1, the remaining segments will use the last color in the list.

©2013-2025 Tencent Cloud International Pte. Ltd.



Property	Description	Type	Required	Note	Supported on iOS	Suppo on Androi
points	Array of latitude and longitude coordinates.	Array	True	[{latitude: 0, longitude: 0}]	True	True
color	Color of the line.	string	False	Hexadecimal.	True	True
colorList	Color of the rainbow lines.	Array	False	Ignore the color value when it exists.	False	Google Maps: False Huawe Maps: False Tence Maps: True Baidu Maps: True Amap: True
width	Width of the line.	number	False		True	True

dottedLine	Whether it is a dotted line.	boolean	False	Default value: false	True	True
arrowLine	Line with arrows.	boolean	False	-	False	True
arrowIconPath	Replaces the arrow icon.	string	False	-	False	True
borderColor	Border color of the line.	string	False	-	False	Google Maps: False Huawe Maps: False Tence Maps: True Baidu Maps: False Amap: False
borderWidth	Width of the border.	number	False	-	False	Google Maps: False Huawe Maps: False Tence Maps: True Baidu Maps: False Amap: False
level	Z-index level for overlapping.	string	False	Default value: abovelabels	False	Google Maps: False Huawe Maps:

						False Tence Maps: True Baidu Maps: False Amap: False
textStyle	Text style.	TextStyle	False	Text style on the polyline.	False	Google Maps: False Huawe Maps: False Tence Maps: True Baidu Maps: False Amap: False
segmentTexts	Segmented texts.	Array<SegmentText>	False	Text content and position for each segment.	False	Google Maps: False Huawe Maps: False Tence Maps: True Baidu Maps: False Amap: False

SegmentText

Property	Description	Type
name	Name.	string

startIndex	Start point.	number
endIndex	End point.	number

TextStyle

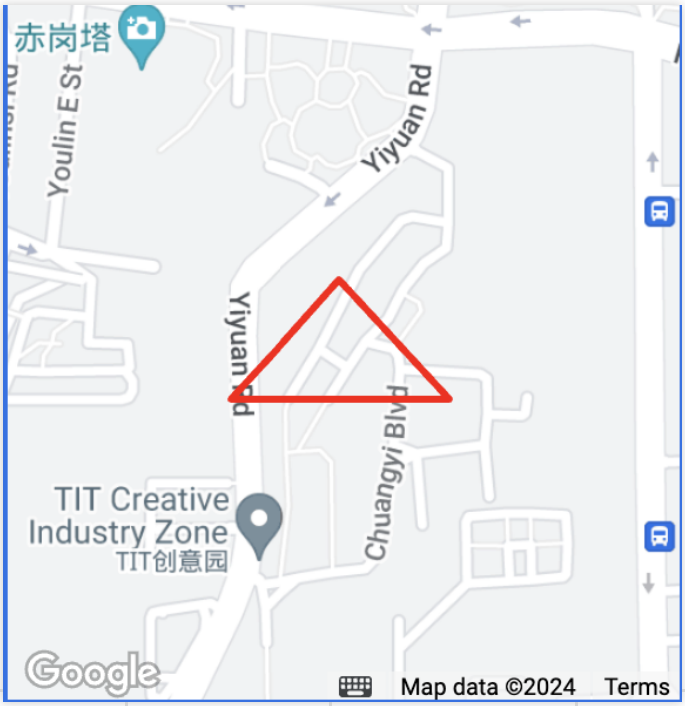
Property	Description	Type	Default value
textColor	Text color.	string	#000000
strokeColor	Stroke color.	number	#ffffff
fontSize	Font size.	number	14

`The level field` indicates the stacking order of map elements. The available options are as follows:

Value	Description
abovelabels	Displays above all POIs.
abovebuildings	Displays above buildings but below POIs.
aboveroads	Displays above roads but below buildings.

polygon

Feature description:A polygon is created by specifying a series of coordinate points, forming a closed shape based on the points data.



Property	Description	Type	Required	Note	Supported on iOS	Supported on Android	
dashArray	Dashed border lines.	Array<number>	False	Default is [0, 0] for a solid line. [10, 10] creates a dashed line with 10 pixels of solid line followed by 10 pixels of space, repeating.	False	Google Maps: False Huawei Maps: False Tencent Maps: True Baidu Maps: False Amap: False	
points	Array of latitude and longitude coordinates.	array	True	[{latitude: 0, longitude: 0}]	True	True	
strokeWidth	Width of the stroke.	number	False	-	True	True	
strokeColor	Color of the	string	False	Hexadecimal	True	True	

	stroke.						
fillColor	Fill color of the polygon.	string	False	Hexadecimal	True	True	
zIndex	Z-axis value of the polygon.	number	False	-	False	True	
level	Stacking order of map elements.	string	False	Default value: abovelabels	False	Google Maps: False Huawei Maps: False Tencent Maps: True Baidu Maps: False Amap: False	

circle

Feature description: Displays circles on the map.



Property	Description	Type	Required	Note	Supported	Supported	Supporte

					on iOS	on Android	in IDE
latitude	The latitude.	number	True	Floating point number. Value range: -90 - 90.	True	True	True
longitude	The longitude.	number	True	Floating points. Value range: -180 - 180.	True	True	True
color	Color of the stroke.	string	False	Floating points. Value range: -180 - 180.	True	True	True
fillColor	Fill color of the circle.	string	False	Hexadecimal	True	True	True
radius	Radius.	number	True	-	True	True	True
strokeWidth	Width of the stroke.	number	False	-	True	True	False
level	Stacking order of map elements.	string	False	Default value: abovelabels	False	Google Maps: False Huawei Maps: False Tencent Maps: True Baidu Maps: False Amap: False	False

control

Feature description: Displays controls on the map that do not move with the map. It is recommended to use cover-view instead since this component will be deprecated soon.

Property	Description	Type	Required	Note	Supported on iOS	Supported on Android	Supported in IDE
id	The control ID.	number	False	This ID will be returned in the callback when the control is tapped.	False	False	False
position	The position of the control on the map.	object	True	The position of the control relative to the map.	False	False	False
iconPath	Icon path.	string	True	Path to the image in the project directory, supports, local path, and code package path.	False	False	False
clickable	Whether the control can be tapped.	boolean	False	Cannot be tapped by default.	False	False	False

position

Property	Description	Type	Required	Note	Supported on iOS	Supported on Android	Supported in IDE
left	Distance from the left edge of the map.	number	False	Default value: 0	False	False	False

top	Distance from the top edge of the map.	number	False	Default value: 0	False	False	False
width	The width.	number	False	It defaults to the width of the image.	False	False	False
height	The height.	number	False	It defaults to the height of the image.	False	False	False

Return values for bindregionchange

Property	Description	Type	Note
type	Triggered when the view changes start or end.	string	begin for start, end for end.
causedBy	Reason for the view change.	string	drag for dragging, scale for scaling, update for API call.

Scale

Zoom level	3	4	5	6	7	8	9	10	11
Scale	1 cm = 1000 km	1 cm = 500 km	1 cm = 200 km	1 cm = 100 km	1 cm = 50 km	1 cm = 25 km	1 cm = 20 km	1 cm = 10 km	1 cm = 5 km
Zoom level	12	13	14	15	16	17	18	19	20
Scale	1 cm = 2 km	1 cm = 1km	1 cm = 500 m	1 cm = 200 m	1 cm = 100 m	1 cm = 50 m	1 cm = 20 m	1 cm = 10 m	1 cm = 5 m

©2013-2025 Tencent Cloud International Pte. Ltd.

©2013-2025 Tencent Cloud International Pte. Ltd.

Media Component

Last updated：2025-05-26 10:52:21

audio

Feature description: Audio component. This component is no longer maintained; it is recommended to use wx.createInnerAudioContext instead.

Parameter and description:

Property	Type	Default value	Required	Description
id	string	-	False	The unique identifier for the audio component.
src	string	-	False	URL of the audio resource to be played.
loop	boolean	false	False	Whether to loop the playback.
controls	boolean	false	False	Whether to show the default controls.
poster	string	-	False	Image URL for the audio cover on the default controls. This is ignored if controls is set to false.
name	string	Unknown song	False	Name of the audio on the default controls. This is ignored if controls is set to false.
author	string	Unknown author	False	Author of the audio on the default controls. This is ignored if controls is set to false.
binderror	eventhandle	-	False	Triggered when an error occurs, detail = {errMsg: MediaError.code}.
bindplay	eventhandle	-	False	Triggered when playback starts or resumes.
bindpause	eventhandle	-	False	Triggered when playback is paused.
bindtimeupdate	eventhandle	-	False	Triggered when the playback progress changes, detail = {currentTime, duration}.
bindended	eventhandle	-	False	Triggered when playback reaches the end.

Example:

WXML

JAVASCRIPT

```
<!-- audio.wxml -->
<audio poster="{{poster}}" name="{{name}}" author="{{author}}" src="{{src}}"
id="myAudio" controls loop></audio>

<button type="primary" bindtap="audioPlay">Playback</button>
<button type="primary" bindtap="audioPause">Pause</button>
<button type="primary" bindtap="audio14">Sets the current playback time to 14
seconds</button>
<button type="primary" bindtap="audioStart">Back to the beginning</button>
```

```
// audio.js
Page({
  onReady: function (e) {
    // Uses wx.createAudioContext to get the audio context
    this.audioCtx = wx.createAudioContext('myAudio')
  },
  data: {
    poster: 'http://y.gtimg.cn/music/photo_new/T002R300x300M000003rsKF44GyaSk.jpg?m
    name: 'At The Moment',
    author: 'Xu Wei',
    src: 'http://ws.stream.qqmusic.qq.com/M500001VfvsJ21xFqb.mp3?guid=ffffffff82def
  },
  audioPlay: function () {
    this.audioCtx.play()
  },
  audioPause: function () {
    this.audioCtx.pause()
  },
  audio14: function () {
    this.audioCtx.seek(14)
  },
  audioStart: function () {
    this.audioCtx.seek(0)
  }
})
```

image

Feature description: Image component.

Parameter and description:

Property	Type	Default	Description

		value	
src	string	-	URL of the image resource.
mode	string	'scaleToFill'	Mode for image cropping and scaling.
show-menu-by-longpress	boolean	false	Shows a menu with options like "Save image" when the image is long-pressed.
lazy-load	boolean	false	Lazy loading of images. Effective only for images within page and scroll-view.
binderror	handleevent	-	Event name published to AppService when an error occurs, event.detail = {errMsg: 'something wrong'}.
bindload	handleevent	-	Event name published to AppService when the image is fully loaded, event.detail = {height:'Image height in px', width:'Image width in px'}.

Notes:

The default width of the image component is 300 px, and the default height is 225 px.

When the image component is scaled, the calculated width and height may include decimals, which might be rounded in different webview kernels.

Valid values for mode: There are 13 modes, including 4 scaling modes and 9 cropping modes.

Mode	Value	Description
Scaling	scaleToFill	Scales the image without preserving the aspect ratio, stretching it to fill the image element completely.
Scaling	aspectFit	Scales the image while preserving the aspect ratio, ensuring the entire image is visible (long side fits).
Scaling	aspectFill	Scales the image while preserving the aspect ratio, ensuring the short side fits, potentially cropping the long side.
Scaling	widthFix	Keeps the width unchanged and adjusts the height automatically, preserving the original aspect ratio.
Cropping	top	Does not scale the image, only shows the top part.
Cropping	bottom	Does not scale the image, only shows the bottom part.
Cropping	center	Does not scale the image, only shows the center part.
Cropping	left	Does not scale the image, only shows the left part.

©2013-2025 Tencent Cloud International Pte. Ltd.

Cropping	right	Does not scale the image, only shows the right part.
Cropping	top left	Does not scale the image, only shows the top-left corner.
Cropping	top right	Does not scale the image, only shows the top-right corner.
Cropping	bottom left	Does not scale the image, only shows the bottom-left corner.
Cropping	bottom right	Does not scale the image, only shows the bottom-right corner.

Example:

WXML

JAVASCRIPT

```
<view class="page">
  <view class="page__hd">
    <text class="page__title">image</text>
    <text class="page__desc">Image</text>
  </view>
  <view class="page__bd">
    <view class="section section_gap" wx:for="{{array}}" wx:for-item="item">
      <view class="section__title">{{item.text}}</view>
      <view class="section__ctn">
        <image style="width: 200px; height: 200px; background-color: #eeeeee;"
mode="{{item.mode}}" src="{{src}}"></image>
      </view>
    </view>
  </view>
</view>
```

```
Page({
  data: {
    array: [{
      mode: 'scaleToFill',
      text: 'scaleToFill: Scales the image without preserving the aspect ratio, str
    }, {
      mode: 'aspectFit',
      text: 'aspectFit: Scales the image while preserving the aspect ratio, ensurin
    }, {
      mode: 'aspectFill',
      text: 'aspectFill: Scales the image while preserving the aspect ratio, ensuri
    }, {
      mode: 'top',
      text: 'top: Does not scale the image, only shows the top part.'
    }, {
```

```
      mode: 'bottom',
      text: 'bottom: Does not scale the image, only shows the bottom part.'
    }, {
      mode: 'center',
      text: 'center: Does not scale the image, only shows the center part.'
    }, {
      mode: 'left',
      text: 'left: Does not scale the image, only shows the left part.'
    }, {
      mode: 'right',
      text: 'right: Does not scale the image, only shows the right part.'
    }, {
      mode: 'top left',
      text: 'top left: Does not scale the image, only shows the top-left part.'
    }, {
      mode: 'top right',
      text: 'top right: Does not scale the image, only shows the top-right part.'
    }, {
      mode: 'bottom left',
      text: 'bottom left: Does not scale the image, only shows the bottom-left part
    }, {
      mode: 'bottom right',
      text: 'bottom right: Does not scale the image, only shows the top-right part.
    }],
    src: 'https://res.wx.qq.com/wxdoc/dist/assets/img/0.4cb08bb4.jpg'
  },
  imageError: function(e) {
    console.log('An error event is triggered in image3, and the carried value is',
  }
})
```

video

Feature description: Video component. The component in an earlier version is a Native Component be aware of the limitations of native components when using it.

Property	Type	Default value	Description
src	string	-	URL of the video resource to be played, supports cloud file ID.
duration	number	-	Specifies the video duration for progress bar（Does not affect the length of the original video）.

controls	boolean	true	Whether to display the default playback controls (play/pause buttons, progress bar, time).
danmu-list	Object Array	-	List of on-screen comments.
danmu-btn	boolean	false	Whether to display the on-screen comment button. This configuration is only effective if set during initialization and cannot be changed later.
enable-danmu	boolean	false	Whether to display the on-screen comments. This configuration is only effective if set during initialization and cannot be changed later.
autoplay	boolean	false	Whether to enable auto-playback.
loop	boolean	false	Whether to loop the playback.
muted	boolean	false	Whether to mute the playback.
initial-time	number	-	Specifies the initial playback position of the video.
page-gesture	boolean	false	Whether to enable brightness and volume adjustment gesture in non-full screen mode.
direction	number	-	Specifies the orientation of the video in full screen mode. If not specified, it is automatically determined based on the aspect ratio. 0 for normal vertical. 90 for counterclockwise 90 degrees. -90 for clockwise 90 degrees.
show-progress	boolean	true	If not specified, the progress will only be displayed when the width is greater than 240.
show-fullscreen-btn	boolean	true	Whether to show the full screen button.
show-play-btn	boolean	true	Whether to show the playback button in the control bar at the bottom of the video.
show-center-play-btn	boolean	true	Whether to show the playback button in the middle of the video.
enable-progress-gesture	boolean	true	Whether to enable the gesture to control progress.
object-fit	string	contain	The presentation format of a video when the size of the video does not match the video container size. contain: Contain.

			fil: Fill. cover: Cover.
poster	string	-	URL for the video cover image.
show-mute-btn	boolean	false	Whether to show the mute button.
title	string	-	Title of the video, displayed at the top in fullscreen.
play-btn-position	string	bottom	The position of the playback button. Valid values: bottom (on the control bar) center (in the center of the video)
enable-play-gesture	boolean	false	Whether to enable the playback gesture, i.e., double tap to switch between playback and pause.
auto-pause-if-navigate	boolean	true	Whether to automatically pause the video on this page when navigating to another mini program page.
auto-pause-if-open-native	boolean	true	Whether to automatically pause the video on this page when navigating to another native page.
vslide-gesture	boolean	false	Whether to enable brightness and volume adjustment gesture in non-full screen mode (same as page-gesture).
vslide-gesture-in-fullscreen	boolean	true	Whether to enable brightness and volume adjustment gesture in full screen mode.（Supported only on IOS）
picture-in-picture-mode	string/Array	-	Sets the system-level small window mode: push, pop, or an array of modes (e.g., ["push", "pop"]).
enable-auto-rotation	boolean	false	Whether to enable automatic fullscreen when the phone is rotated, effective when system auto-rotate is enabled.
bindplay	eventhandle	-	Triggered when playback starts or resumes.
bindpause	eventhandle	-	Triggered when playback is paused.
bindended	eventhandle	-	Triggered when playback reaches the end.
bindtimeupdate	eventhandle	-	Triggered when playback progress changes, event.detail = {currentTime, duration}. Trigger frequency: 250 ms.
bindfullscreenchange	eventhandle	-	Triggered when the video enters/exits full screen

©2013-2025 Tencent Cloud International Pte. Ltd.

			mode; event.detail = {fullScreen, direction}. Valid values: vertical, horizontal.
bindwaiting	eventhandle	-	Triggered when the video is buffering.
binderror	eventhandle	-	Triggered when an error occurs during video playback.
bindprogress	eventhandle	-	Triggered when the loading progress changes. supports only one segment of loading. event.detail = {buffered},in percentage.
bindloadedmetadata	eventhandle	-	Triggered when the video metadata is loaded. event.detail = {width, height, duration}
bindseekcomplete	eventhandle	-	Triggered when seek is complete (position in seconds for iOS, in milliseconds for Android).

Notes:

<video>The default width is 300 px and the default height is 225 px. You can set the width and height using wxss.

Picture-in-picture feature description:

The video component supports the following three trigger modes for picture-in-picture (PiP) (set via the picture-in-picture-mode property):

push mode: PiP appears when navigating from the current page to the next page (page stack push).

pop mode: PiP is triggered when leaving the current page (page stack pop).

Both of these routing behaviors will trigger PiP.

Additionally, PiP supports the following features:

The size of the PiP container is automatically determined based on the original component size.

Tapping on the PiP will navigate the user back to the page containing the corresponding video player.

After PiP appears, users can close it by tapping the close button in the top right corner of the PiP or by calling the context.exitPictureInPicture() method.

When the player enters PiP mode, the page containing the player will enter the hide state (triggering the onHide lifecycle event) but will not be destroyed. When the PiP is closed, the page containing the player will be unloaded (triggering the onUnload lifecycle event).

Example:

WXML

JAVASCRIPT

```
<view class="page-body">
  <view class="page-section tc">
    <video
      id="myVideo"
```

```
      src="http://wxsnsdy.tc.qq.com/105/20210/snsdyvideodownload?
filekey=30280201010421301f0201690402534804102ca905ce620b1241b726bc41dcff44e0020
4012882540400&bizid=1023&hy=SH&fileparam=302c020101042530230204136ffd93020457e3
c4ff02024ef202031e8d7f02030f42400204045a320a0201000400"
      binderror="videoErrorCallback"
      danmu-list="{{danmuList}}"
      enable-danmu
      danmu-btn
      show-center-play-btn='{{false}}'
      show-play-btn="{{true}}"
      controls
      picture-in-picture-mode="{{['push', 'pop']}}"
      bindenterpictureinpicture='bindVideoEnterPictureInPicture'
      bindleavepictureinpicture='bindVideoLeavePictureInPicture'
    ></video>
    <view style="margin: 30rpx auto" class="weui-label">On-screen comment
content</view>
    <input bindblur="bindInputBlur" class="weui-input" type="text"
placeholder="Type your on-screen comment here" />
    <button style="margin: 30rpx auto"  bindtap="bindSendDanmu" class="page-
body-button" type="primary" formType="submit">Send on-screen comment</button>
    <navigator style="margin: 30rpx auto"  url="picture-in-picture" hover-
class="other-navigator-hover">
      <button type="primary" class="page-body-button"
bindtap="bindPlayVideo">Picture-in-picture mode</button>
    </navigator>
  </view>
</view>
```

```
function getRandomColor() {
  const 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({
  onShareAppMessage() {
    return {
      title: 'video',
      path: 'page/component/pages/video/video'
    }
  },
```

```
onReady() {
  this.videoContext = wx.createVideoContext('myVideo')
},

onHide() {

},

inputValue: '',
data: {
  src: '',
  danmuList:
  [{
    text: 'The on-screen comment that appears in the first second',
    color: '#ff0000',
    time: 1
  }, {
    text: 'The on-screen comment that appears in the third second',
    color: '#ff00ff',
    time: 3
  }],
},

bindInputBlur(e) {
  this.inputValue = e.detail.value
},

bindButtonTap() {
  const that = this
  wx.chooseVideo({
    sourceType: ['album', 'camera'],
    maxDuration: 60,
    camera: ['front', 'back'],
    success(res) {
      that.setData({
        src: res.tempFilePath
      })
    }
  })
},

bindVideoEnterPictureInPicture() {
  console.log('Enters picture-in-picture mode')
},

bindVideoLeavePictureInPicture() {
```

```
    console.log ('Exits picture-in-picture mode')
  },

  bindPlayVideo() {
    console.log('1')
    this.videoContext.play()
  },
  bindSendDanmu() {
    this.videoContext.sendDanmu({
      text: this.inputValue,
      color: getRandomColor()
    })
  },

  videoErrorCallback(e) {
    console.log ('Video error message: ')
    console.log(e.detail.errMsg)
  }
})
```

animation-video

Feature description: The animation-video component is a frontend component that allows mini programs to render specific video resources with a transparent background, enabling developers to create immersive and rich animation effects at a low cost. The animation-video component also provides a variety of APIs to control the animation's playback, pause, jump to a specified position, and more. For related APIs, see wx.createAnimationVideoContext.
Parameter and description:

Property	Type	Default value	Required	Description
resource-width	number	800	False	The width of the video resource used by the component (unit: px).
resource-height	number	400	False	The height of the video resource used by the component (unit: px).
canvas-style	string	width:400px; height:400px;	False	CSS styles for animation canvas.
path	string	-	True	URL of the animation resource, supports relative and remote paths. If it is a remote path,**Access-Control-Allow-Origin needs to**

				be set in the response header to prevent cross-origin issues.
loop	boolean	false	False	Whether to loop the animation playback.
autoplay	boolean	false	False	Whether to enable auto-playback for the animations.
alpha-direction	string	left	False	Direction of the alpha channel in the video resource. left means the alpha channel is on the left side, right means it is on the right side.
bindstarted	eventhandle	-	False	The callback is triggered when the animation starts playing.
bindended	eventhandle	-	False	Event triggered when the animation reaches the end (natural end of playback triggers the callback, loop end and pause do not trigger it).

Example:

WXML

JAVASCRIPT

```
<view class="wrap">
    <view class="card-area">
        <view class="top-description border-bottom">
            <view>Animation alpha channel information is on the right side of
the video</view>
            <view>alpha-direction='right'</view>
        </view>
        <view class="video-area">
            <animation-video
                path="{{rightAlphaSrcPath}}"
                loop="{{loop}}"
                resource-width="800"
                resource-height="400"
                canvas-style="width:200px;height:200px"
                autoplay="{{autoplay}}"
                bindstarted="onStarted"
                bindended="onEnded"
                alpha-direction='right'
            ></animation-video>
        </view>
    </view>
```

```
    <view class="card-area">
        <view class="top-description border-bottom">
            <view>Animation alpha channel information is on the left side of
the video</view>
            <view>alpha-direction='left'</view>
        </view>
        <view class="video-area">
            <animation-video
                path="{{leftAlphaSrcPath}}"
                loop="{{loop}}"
                resource-width="800"
                resource-height="400"
                canvas-style="width:200px;height:200px"
                autoplay="{{autoplay}}"
                bindstarted="onStarted"
                bindended="onEnded"
                alpha-direction='left'
            ></animation-video>
        </view>
    </view>
</view>
```

```
 Page({
  data: {
    loop: true,
    leftAlphaSrcPath: 'https://efe-h2.cdn.bcebos.com/ceug/resource/res/2020-1/15779
    rightAlphaSrcPath: 'https://b.bdstatic.com/miniapp/assets/docs/alpha-right-exam
    autoplay: true
  },
  onStarted() {
    console.log('===onStarted');
  },
  onEnded() {
    console.log('===onEnded');
  }
})
```

camera

Feature description: System camera. This component is a Native Component and there are some restrictions for using it. For related APIs, see wx.createCameraContext.

Parameter and description:

Property	Type	Valid values	Default value	Description
mode	string	normal: Camera mode ScanCode: Scan code mode	normal	Application mode. This configuration is only effective if set during initialization and cannot be changed later.
device-position	string	front: Front-facing back: Rear-facing	back	Camera position.
flash	string	auto: Auto on: Enabled off: Disabled torch: Always on	auto	Flash. Valid values: auto, on, off.
bindstop	eventhandle	-	-	Triggered when the camera stops abnormally, such as exiting the background.
binderror	eventhandle	-	-	Triggered when the user denies camera access.
bindscancode	eventhandle	-	-	Triggered when a scan code is successfully recognized, only effective when mode="scanCode".

Notes:

User authorization is required for scope.camera.

Only one camera component can be inserted to a page.

The onCameraFrame API returns raw frame data of different sizes based on frame-size, which may differ from the image displayed by the camera component. The actual pixel values are determined by the system.

Example:

WXML

JAVASCRIPT

```
<camera
  device-position="back"
  flash="off"
  binderror="error"
  style="width: 100%; height: 300px;"
></camera>
```

```
<button type="primary" bindtap="takePhoto">Camera</button>
<view>Preview</view>
<image mode="widthFix" src="{{src}}"></image>
```

```
Page({
  takePhoto() {
    const ctx = wx.createCameraContext()
    ctx.takePhoto({
      quality: 'high',
      success: (res) => {
        this.setData({
          src: res.tempImagePath
        })
      }
    })
  },
  error(e) {
    console.log(e.detail)
  }
})
```

live-player

Feature description: Real-time playback of audios and videos. This component is a native component, and there are some restrictions for using it. For details, see Native Components.

Parameter and description:

Property	Type	Valid values	Default value	Description
src	string	-	-	URL of the audios and videos. Only flv and rtmp formats are supported.
mode	string	live: Live streaming RTC: Real-time call	live	The mode.
autoplay	boolean	-	false	Auto-playback.
muted	boolean	-	false	Whether the audios and videos are muted.

orientation	string	vertical: Portrait horizontal: Landscape	vertical	Screen orientation.
object-fit	string	contain: Image's long side fills the screen, short side areas are black. fillCrop: Image fills the screen, excess parts are cropped.	contain	Fill mode. Valid values: contain, fillCrop.
min-cache	number	-	1	Minimum buffer size in seconds (0.2 seconds is recommended for the RTC mode).
max-cache	number	-	3	Maximum buffer size in seconds (0.8 seconds is recommended for the RTC mode). The buffer is used to resist network fluctuations. Larger buffer improves network stability but increases latency.
sound-mode	string	speaker: Speaker ear: Earphone	speaker	Sound output mode.
auto-pause-if-navigate	boolean	-	true	Whether to automatically pause the live audio and video on this page when navigating to another mini program page.
auto-pause-if-open-native	boolean	-	true	Whether to automatically pause the live audio and video on this page when navigating to another native page of the host app.
picture-in-picture-mode	string/Array	[]: Disable the	-	Sets the picture-in-picture

		picture-in-picture. push: Trigger the picture-in-picture when routing push is performed pop: Trigger the picture-in-picture when routing pop is performed		mode: push, pop, or an array of modes (e.g., ["push", "pop"]).
enable-auto-rotation	boolean	-	false	Whether to enable automatic full-screen when the phone is rotated, effective when system auto-rotate is enabled.
enable-metadata	boolean	-	false	Whether to call back metadata.
bindstatechange	eventhandle	-	-	Triggered when playback state changes, detail = {code}.
bindfullscreenchange	eventhandle	-	-	Triggered when full-screen mode changes, detail = {direction, fullScreen}.
bindnetstatus	eventhandle	-	-	Network status notification, detail = {info}.
bindaudiovolumenotify	eventhandle	-	-	Playback volume notification, detail = {}.
bindenterpictureinpicture	eventhandle	-	-	Triggered when the player enters PiP mode.
bindleavepictureinpicture	eventhandle	-	-	Triggered when the player exits PiP mode.
bindmetadatachange	eventhandle	-	-	Metadata notification, detail = {info}.

Status code

Code	Description

2001	Connected to the server successfully.
2002	Connected to the server and started the stream pulling.
2003	The network received the first video data packet (IDR).
2004	Video playback started.
2006	Video playback ended.
2007	Video playback is loading.
2008	Decoder started.
2009	Video resolution changed.
-2301	The network is disconnected, and automatic reconnection fails multiple times. Please restart the playback manually.
-2302	Failed to get accelerated stream URL.
2101	Failed to decode the current video frame.
2102	Failed to decode the current audio frame.
2103	The network is disconnected, and automatic reconnection has been started.
2104	Unstable incoming network packets. This may be caused by insufficient downstream bandwidth, or uneven outbound traffic from the source.
2105	Video freezes occurred during playback.
2106	Hardware decoding failed, switching to software decoding.
2107	Current video frames are not continuous, possible frame loss.
2108	The first I-frame hardware decoding failed, SDK automatically switches to software decoding.
3001	RTMP - DNS resolution failed.
3002	RTMP server connection failed.
3003	RTMP server handshake failed.
3005	RTMP read/write failed.

©2013-2025 Tencent Cloud International Pte. Ltd.

Network status data

Key name	Description
videoBitrate	Current video data reception bit rate (in kbps).
audioBitrate	Current audio data reception bit rate (in kbps).
videoFPS	Current video frame rate.
videoGOP	Current video GOP (Group of pictures), i.e., the time interval between every two keyframes (I frames) (unit: s).
netSpeed	Current sending/receiving speed.
netJitter	Network jitter, with higher values indicating less stable network conditions.
videoWidth	Width of the video image.
videoHeight	Height of the video image.

Notes:

The default width of live-player is 300 px and the default height is 225 px. You can set the width and height using wxss.

This component is not supported on developer tools.

Picture-in-picture feature description:

The video component supports the following three trigger modes for picture-in-picture (PiP) (set via the picture-in-picture-mode property):

push mode: PiP appears when navigating from the current page to the next page (page stack push).

pop mode: PiP is triggered when leaving the current page (page stack pop).

Both of these routing behaviors will trigger PiP.

Additionally, PiP supports the following features:

The size of the PiP container is automatically determined based on the original component size.

Tapping on the PiP will navigate the user back to the page containing the corresponding video player.

After PiP appears, users can close it by tapping the close button in the top right corner of the PiP or by calling the context.exitPictureInPicture() method.

When the player enters PiP mode, the page containing the player will enter the hide state (triggering the onHide lifecycle event) but will not be destroyed. When the PiP is closed, the page containing the player will be unloaded (triggering the onUnload lifecycle event).

Example:

WXML

JAVASCRIPT

```
<live-player
    src="https://domain/pull_stream"
    mode="RTC"
    autoplay
    bindstatechange="statechange"
    binderror="error"
    style="width: 300px; height: 225px;"
/>
```

```
Page({
  statechange(e) {
    console.log('live-player code:', e.detail.code)
  },
  error(e) {
    console.error('live-player error:', e.detail.errMsg)
  }
})
```

live-pusher

Feature description: The live-pusher component is used for real-time audio and video recording (same-layer rendering is not supported).

Parameter and description:

Property	Type	Valid values	Default value	Description
url	string	-	-	Stream URL. Only RTMP is supported.
mode	string	-	RTC	SD (Standard Definition), HD (High Definition), FHD (Full High Definition), RTC (Real-Time Communication).
autopush	boolean	-	false	Automatic stream pushing.
muted	boolean	-	false	Whether the audios and videos are muted.
enable-camera	boolean	-	true	Enable the camera.
auto-focus	boolean	-	true	Auto focus.

orientation	string	vertical: Portrait horizontal: Landscape	vertical	Screen orientation.
beauty	number	-	0	Beauty filter. Value range: 0-9, where 0 indicates off.
whiteness	number	-	0	Whiteness filter. Value range: 0-9, where 0 indicates off.
aspect	string	-	9：16	Aspect ratio. Valid values: 3:4, 9:16.
min-bitrate	number	-	200	Min bitrate.
max-bitrate	number	-	1000	Max bitrate.
audio-quality	string	-	height	High quality (48 kHz) or low quality (16 kHz). Valid values: high, low.
waiting-image	string	-	-	Waiting image when the app goes to the background.
waiting-image-hash	string	-	-	The MD5 value of the waiting image resource.
zoom	boolean	-	false	Adjusts the zoom.
device-position	string	-	-	Camera position. Valid values: front, back.
background-mute	boolean	-	false	Whether to mute the sound when entering the background.
mirror	boolean	-	false	Whether mirror the push stream, reflected in the live-player.
local-mirror	string	auto: Front camera mirrors, back camera does not enable: Both front and rear cameras mirror	auto	Controls local preview mirroring.

©2013-2025 Tencent Cloud International Pte. Ltd.

		disable: Neither camera mirrors		
audio-reverb-type	number	0: Disable 1: KTV 2: Small room 3: Auditorium 4: Deep 5: Loud 6: Metal 7: Magnetic	0	Types of audio reverberation.
enable-mic	boolean	-	-	Enable or disable the microphone.
enable-agc	boolean	-	false	Whether to enable audio automatic gain control (AGC).
enable-ans	boolean	-	false	Whether to enable audio noise suppression.
audio-volume-type	string	auto: Auto media: Media volume voicecall: Call volume	auto	Volume type.
video-width	number	-	360	Resolution width of the pushed video stream.
video-height	number	-	640	Resolution height of the pushed video stream.
bindstatechange	eventhandle	-	-	Status change event, detail = {code}.
bindnetstatus	eventhandle	-	-	Network status notification, detail = {info}.
binderror	eventhandle	-	-	Render error event, detail = {errMsg, errCode}.
bindbgmstart	eventhandle	-	-	Triggered when the background sound starts playing.
bindbgmprogress	eventhandle	-	-	Triggered when the background sound progress changes, detail

©2013-2025 Tencent Cloud International Pte. Ltd.

				= {progress, duration}.
bindbgmcomplete	eventhandle	-	-	Triggered when the background sound finished playing.

Notes:

User authorization is required for scope.camera and scope.record.

This component is not supported on developer tools.

The default width of live-pusher is 100%, and there is no default height. Please set the width and height using wxss.

Error code

Code	Description
10001	User has denied access to the camera.
10002	User has denied access to the microphone.
10003	Failed to load the background music (BGM) resource.
10004	Failed to load the waiting image resource (waiting-image).

Status code

Code	Description
1001	Connected to the stream pushing server.
1002	Completed handshake with the server and started stream pushing.
1003	Enabled the camera successfully.
1004	Started screen recording successfully.
1005	Dynamic resolution adjustment during streaming.
1006	Dynamic bitrate adjustment during streaming.
1007	Captured the first frame successfully.
1008	Started the encoder successfully.
-1301	Failed to enable the camera.
-1302	Failed to enable the microphone.
-1303	Failed to encode the video.

©2013-2025 Tencent Cloud International Pte. Ltd.

-1304	Failed to encode the audio.
-1305	Unsupported video resolution.
-1306	Unsupported audio sample rate.
-1307	The network is disconnected, and automatic reconnection fails multiple times. Please restart the stream pushing manually.
-1308	Failed to start the screen recording. Possible reason: Rejected by the user.
-1309	Screen recording failed. Reason: Unsupported Android version (5.0 or above version is required).
-1310	Screen recording is interrupted by another application.
-1311	Enabled Android microphone successfully, but no audio data captured.
-1312	Failed to switch between landscape and portrait screens during screen recording.
1101	Poor network conditions: Insufficient uplink bandwidth, data upload obstructed.
1102	The network is disconnected, and automatic reconnection has been started.
1103	Hardware decoding failed, switching to software decoding.
1104	Failed to encode the video.
3001	RTMP - DNS resolution failed.
3002	RTMP server connection failed.
3003	RTMP server handshake failed.
3004	RTMP server disconnected, please check the validity of the stream URL or anti-leech link expiration.
3005	RTMP read/write failed.

Network status data

Key name	Description
videoBitrate	Current video encoder output bit rate (in kbps).
audioBitrate	Current audio encoder output bit rate (in kbps).

videoFPS	Current video frame rate.
videoGOP	Current video GOP, i.e., the time interval between every two keyframes (I frames) (unit: s).
netSpeed	Current sending/receiving speed.
netJitter	Network jitter, with higher values indicating less stable network conditions.
videoWidth	Width of the video image.
videoHeight	Height of the video image.

Example:

WXML

JAVASCRIPT

```
<live-pusher
    url="https://domain/push_stream"
    mode="RTC"
    autopush
    bindstatechange="statechange"
    style="width: 300px; height: 225px;"
/>
```

```
Page({
  statechange(e) {
    console.log('live-pusher code:', e.detail.code)
  }
})
```

Open Capabilities

Last updated：2025-04-29 18:40:51

ad

Feature description: Banner ad.

Parameter and description: The prerequisite for using the mini program ad feature is that commercialization permissions have been enabled.

Property	Type	Default value	Description
unit-id	string	-	Ad unit ID, which can be created in the ad module of the console. Required.
bindload	eventhandler	-	Callback for successful ad load.
binderror	eventhandler	-	Callback for ad load failure, event.detail = {errCode: 1002}

Error code

Error codes are obtained from the error information. During debugging, you can capture error details via exception returns. You can add appropriate monitoring information to assist in troubleshooting issues in the live environment.

Error code	Description	Reason
1000	Failed to call the backend API	This error is not caused by the developer.
1001	Parameter error	Incorrect usage.
1002	Invalid ad unit	Incorrect ad unit ID spelling or accidental use of another app's ad unit ID.
1003	Internal error	This error is not caused by the developer.
1004	No suitable ad	Ads do not appear every time; this time it might not appear because there are no ads that match the user's preferences.
1010	Ad feature disabled	The ad feature has been disabled.

web-view

Feature description:A container that hosts the HTML5-based web pages. It will automatically fill the mini program page.

Parameter and description:

Property	Type	Default value	Description
src	string	-	The URL of the HTML5-based web page that the web-view component points to. Configure the HTML5-based web page domain in the mini program management backend. When loading an offline HTML5-based web page, the src should start with file:// or / (pointing to an HTML page, e.g., file://xxx/index.html). You need to configure the staticPath property (an array of strings) in app.json, e.g. ["xxx"].
use-web-title	boolean	true	The title of the navigation bar of the page where the web-view component is located, whether or not to display the title of the H5 page.
bindmessage	eventhandler	-	Triggered when the HTML5-based web page uses the wx.miniProgram.postMessage to send a message to the mini program. This occurs during back navigation, component destruction, or sharing. e.detail = { data }, where data is an array of parameters from multiple postMessage calls.
bindload	eventhandler	-	Triggered when the HTML5-based web page is loaded successfully. e.detail = { src }
binderror	eventhandler	-	Triggered when the HTML5-based web page fails to load. e.detail = { url, fullUrl }, where fullUrl is the complete URL that failed to load.
bindevent	eventhandler	-	Triggered when the HTML5-based web page uses the wx.miniProgram.sendWebviewEvent to send a message to the mini program. e.detail = { data }, where data is the parameter from sendWebviewEvent.

Notes:

The iframe domain within the HTML5-based web page must also be configured as the allowed domain.

you can click the debug button on the <web-view> component to open debugging for the <web-view> component in the mini program IDE.

Each HTML5-based web page can only have one <web-view>. It will automatically fill the entire page and cover other components.

©2013-2025 Tencent Cloud International Pte. Ltd.

Communication between the web page in <web-view> and the mini program is not supported except for the APIs provided by JSSDK.
Avoid using special characters (except a-z,A-Z, 0-9) in the URL, as it may cause a white screen issue on iOS. It is recommended to use encodeURIComponent to escape the URL parameters.
For details, see JSSDK API description.
The priority of the navigation bar title on the page containing the <web-view> component is as follows: `<title></title>` (HTML5 with title configured and use-web-title is true) > `navigationBarTitleText` (page.json)> `navigationBarTitleText` (app.json)
When specifying the static resource address for src, it must start with file:// or / and point to a specific HTML page.
The static resources must be located in the root directory of the mini program. Additionally, you need to configure the staticPath property in app.json (an array of strings representing the directory names of the static resources). Multiple static resources can be configured, but they are only supported within the main package. When uploading the mini program, all static resources are uploaded together with the mini program, and the total size must not exceed the size limit for the entire package.

Authorization

Some APIs require user consent before they can be called. These APIs are divided into multiple scopes based on their usage range. Users can choose to authorize a specific scope, and once authorized, all APIs within that scope can be used directly.
When calling such APIs:
If the user has neither accepted nor declined this permission, a pop-up will appear asking for their consent. The API can only be called if the user agrees.
If the user has already granted permission, the APIs can be called directly.
If the user has rejected authorization, no pop-up will appear, and the API will immediately trigger the fail callback.**Developers should ensure their applications handle scenarios where the user has denied permission**.

Get user authorization status

Developers can get the user's current authorization status using wx.getSetting.

Open the settings page

Users can control the authorization status of the mini program in the settings page (accessible via the "..." menu in the top right corner).
Developers can call wx.openSetting to open the settings page and guide users to enable the authorization.

Initiate an authorization request in advance

Developers can use wx.authorize to initiate authorization requests to users before calling the APIs that require authorization.

Scope list

Scope	APIs	Description
scope.userinfo	wx.getUserInfo	User information.
scope.userLocation	wx.getLocation、wx.startLocationUpdate、wx.startLocationUpdateBackground	Gets geographic location information.
scope.userFuzzyLocation	wx.getFuzzyLocation	Gets approximate geographic location information.
scope.record	live-pusher component or wx.startRecord, wx.operateRecorder	Live streaming or recording.
scope.camera	Camera component	Camera component.
scope.addPhoneCalendar	wx.addPhoneCalendar	Adds logs.
scope.writePhotosAlbum	wx.saveImageToPhotosAlbum	Saves images to an album.
scope.bluetooth	wx.openBluetoothAdapter	Bluetooth.
scope.chooseImage	wx.chooseImage	Selects images in the album.
scope.chooseVideo	wx.chooseVideo	Selects videos in the album.
scope.chooseMedia	wx.chooseMedia	Selects images or videos from the album.
scope.makeMedia	wx.makeMedia	Takes photos or videos.
scope.addPhoneContact	wx.addPhoneContact、wx.chooseContact	Adds/selects contacts.

Authorization validity

Once a user explicitly agrees to or rejects an authorization, the authorization status is recorded locally on the client until the mini program is deleted.

Practice guide

Request authorization from the user only when you actually need to use the authorized API. Clearly explain the reason for needing the functionality in the authorization request.

©2013-2025 Tencent Cloud International Pte. Ltd.

Canvas

Last updated：2024-11-27 18:14:43

Feature description: Canvas. This component is a native component, and there are some restrictions when using it. The related API is wx.createCanvasContext. Currently the new Canvas 2D API is supported, and the old Canvas API is no longer updated and supported.

Parameter and description:

Property	Type	Default value	Description
type	string	2d	Supported canvas type: 2d only.
canvas-id	string	-	Unique identifier of the canvas component.
disable-scroll	boolean	false	Prevents screen scrolling and pull-to-refresh when moving within the canvas and gesture events are bound.
bindtouchstart	eventhandle	-	Triggered when a finger touch action starts.
bindtouchmove	eventhandle	-	Triggered when a finger moves after touching.
bindtouchend	eventhandle	-	Triggered when a finger touch action ends.
bindtouchcancel	eventhandle	-	Triggered when a finger touch action is interrupted, such as by an incoming call or popup.
bindlongtap	eventhandle	-	Triggered after a finger long-presses for 500 ms; moving after triggering the long-press event will not cause the screen to scroll.
binderror	eventhandle	-	Triggered when an error occurs. detail = {errMsg: 'something wrong'}

Notes:

The default width and height of the canvas tag are 300 pixels and 150 pixels respectively.

The canvas-id must be unique within the same page; if a duplicate canvas-id is used, the corresponding canvas will be hidden and will not function properly.

Avoid setting excessively large width and height values, as this can cause crashes on Android devices.

The old canvas API uses the canvas-id to uniquely identify the canvas, and the new Canvas 2D API can directly use the id for identification.

The old canvas API uses wx.createCanvasContext to synchronously obtain the CanvasContext. In contrast, the new Canvas 2D interface requires using SelectorQuery to asynchronously obtain the Canvas object, and then

Canvas.getContext to get the RenderingContext.
The canvas size in the old canvas API is determined by the actual rendering width and cannot be modified by developers. The new Canvas 2D API allows developers to modify the logical size of the canvas, with the default width and height of 300 pixels and150 pixels respectively. The physical pixels and logical pixels may not be equal on different devices, so you need to use wx.getWindowInfo to obtain the device's pixel ratio and multiply it by the actual size of the canvas.

Example (new API):

WXML

JAVASCRIPT

```
<canvas type="2d" id="myCanvas"></canvas>
```

```
Page({
  onReady() {
    const query = wx.createSelectorQuery()
    query.select('#myCanvas')
      .fields({ node: true, size: true })
      .exec((res) => {
        const canvas = res[0].node
        const ctx = canvas.getContext('2d')

        const dpr = wx.getSystemInfoSync().pixelRatio
        canvas.width = res[0].width * dpr
        canvas.height = res[0].height * dpr
        ctx.scale(dpr, dpr)

        ctx.fillRect(0, 0, 100, 100)
      })
  }
})
```

Example (old API):

WXML

JAVASCRIPT

```
<!-- canvas.wxml -->
<canvas style="width: 300px; height: 200px;" canvas-id="firstCanvas"></canvas>
<!-- When using absolute positioning, the canvas at the back of the document
flow is displayed at a higher level than the canvas at the front -->
<canvas style="width: 400px; height: 500px;" canvas-id="secondCanvas"></canvas>
<!-- Because the canvas-id is the same as the previous canvas, the canvas will
not be displayed and an error event will be sent to AppService -->
<canvas style="width: 400px; height: 500px;" canvas-id="secondCanvas"
binderror="canvasIdErrorCallback"></canvas>
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
Page({
  canvasIdErrorCallback: function (e) {
    console.error(e.detail.errMsg)
  },
  onReady: function (e) {
    // Uses wx.createContext to get the Canvas context
    var context = wx.createCanvasContext('firstCanvas')

    context.setStrokeStyle("#00ff00")
    context.setLineWidth(5)
    context.rect(0, 0, 200, 200)
    context.stroke()
    context.setStrokeStyle("#ff0000")
    context.setLineWidth(2)
    context.moveTo(160, 100)
    context.arc(100, 100, 60, 0, 2 * Math.PI, true)
    context.moveTo(140, 100)
    context.arc(100, 100, 40, 0, Math.PI, false)
    context.moveTo(85, 80)
    context.arc(80, 80, 5, 0, 2 * Math.PI, true)
    context.moveTo(125, 80)
    context.arc(120, 80, 5, 0, 2 * Math.PI, true)
    context.stroke()
    context.draw()
  }
})
```

Navigation Component

Last updated：2025-05-22 17:11:58

navigator

Feature description: Page URL.

Parameter and description:

Property	Type	Valid values and description	Default value	Required	Description
target	string	self: The current mini program miniProgram: Other mini programs	self	False	Specifies the target for th the current mini program
url	string	-	-	False	URL for navigation within program.
open-type	string	navigate: Corresponds to the feature of wx.navigateTo or wx.navigateToMiniProgram redirect: Corresponds to the feature of wx.redirectTo switchTab: Corresponds to the feature of wx.switchTab reLaunch: Corresponds to the feature of wx.reLaunch navigateBack: Corresponds to the feature of wx.navigateBack exit: Exits the mini program. Effective when target="miniProgram"	navigate	False	Specifies the navigation
delta	number	-	1	False	Effective when open-type indicates the number of l
app-id	string	-	-	False	Effective when target="m type="navigate", specifie program to open.
path	string	-	-	False	Effective when target="m

					the page path to open. If page.
extra-data	object	-	-	False	Effective when target="m data to pass to the target target mini program can in `App.onLaunch()` a
version	string	develop: The development version trial: The Preview release: The released version. This property is valid only when the version of the mini program is the development version or Preview. If the current mini program version is released, the version of the mini program to be opened must be the released version.	release	False	Effective when target="m type="navigate", specifie program to open.
hover-class	string	-	navigator-hover	False	Specifies the style class If `hover-class="non` effect.
hover-stop-propagation	boolean	-	false	False	Specifies whether to prev from showing the tap effe
hover-start-time	number	-	50	False	Time in milliseconds bef appears after pressing.
hover-stay-time	number	-	600	False	Time in milliseconds the the finger is released.
bindsuccess	string	-	-	False	Effective when target="m when navigation to the m successful.
bindfail	string	-	-	False	Effective when target="m when navigation to the m
bindcomplete	string	-	-	False	Effective when target="m when navigation to the m complete.

Restrictions: User confirmation is required for navigation.

Before navigating to another mini program, a pop-up will appear asking for user confirmation. Navigation will proceed only if the user confirms. If the user cancels, the `fail cancel` will be triggered.

Notes:

`navigator-hover` defaults to `{background-color: rgba(0, 0, 0, 0.1); opacity: 0.7;},`

`The background color of the child nodes of <navigator>` should be transparent.

Example:

sample.wxml

navigator.wxml

redirect.wxml

JAVASCRIPT

WXSS

```
<view class="btn-area">
  <navigator
    url="/page/navigate/navigate?title=navigate"
    hover-class="navigator-hover"
  >
    Navigate to a new page.
  </navigator>
  <navigator
    url="../../redirect/redirect/redirect?title=redirect"
    open-type="redirect"
    hover-class="other-navigator-hover"
  >
    Open on the current page.
  </navigator>
</view>
```

```
<view style="text-align:center"> {{title}} </view>
<view> Tap the top left corner to return to the previous page. </view>
```

```
<view style="text-align:center"> {{title}} </view>
<view> Tap the top left corner to return to the parent page. </view>
```

```
Page({
  onLoad: function(options) {
    this.setData({
      title: options.title
    })
  }
})
```

```
/** wxss **/
/** Modify the default tap effect of navigator. **/
.navigator-hover {
  color: blue;
}
/** Customize the style classes for other tap effect. **/
.other-navigator-hover {
  color: red;
}
```

Form Component

Last updated：2025-05-26 10:53:38

button

Feature description: Button.

Parameter and description:

Property	Type	Valid values and description	Default value	Required	Description
size	string	default: Default size mini: Small size	default	False	Button size.
type	string	primary: Green default: White warn: Red	default	False	Button style type.
plain	boolean	-	false	False	Whether the button is hollow with a background color.
disabled	boolean	-	false	False	Whether the button is disabled.
loading	boolean	-	false	False	Whether to show a loading icon before th button name.
form-type	string	submit: Submits the form reset: Resets the form	-	False	Used for the form component. Triggers the form component's submit/reset event when tapped
open-type	string	share: Triggers user sharing getPhoneNumber (Only supported on SaaS): Gets the user's phone number from the bindgetphonenumber callback	-	False	Host app capabilities

		getEmailAddress (Only supported on SaaS): Gets the user's email address from the bindgetemailaddress callback getUserInfo: Gets user information from the bindgetuserinfo callback openSetting: Opens the authorization settings page feedback: Opens the Feedback page, where users can submit feedback and upload logs chooseAvatar (Only supported on SaaS): Gets the user's profile photo from the bindchooseavatar callback			
hover-class	string	-	button-hover	False	Specifies the style cla when the button is tapped. When `hover-class="none"` is set, no tap effect is available.
hover-start-time	number	-	20	False	Time in milliseconds before the tap effect appears after pressin
hover-stay-time	number	-	70	False	Time in milliseconds t tap effect remains aft the finger is released.
lang	string	en: English zh_CN: Simplified Chinese	en	False	Specifies the languag for returning user information.

		zh_TW: Traditional Chinese			
bindgetuserinfo	eventhandle	-	-	False	When this button is tapped, the acquired user information will b returned. The callbac detail is consistent wi that returned by wx.getUserInfo. Effective when open-type="getUserInfo".
bindgetphonenumber	eventhandle	-	-	False	Callback for quick verification of phone number. Effective whe open-type=getPhoneNumb **Note: After triggerir the bindgetphonenumb callback, immediate hide the phone number button component or set it to disabled to avoic extra charges from repeated authorizations.**
bindopensetting	eventhandle	-	-	False	Callback after openin the authorization setti page. Effective when open-type=openSetti

Notes:

Currently, a button with form-type will only be effective for the form within the current component. Therefore, encapsulating the button in a custom component and the form outside the custom component will make the form-type of the button invalid.

Example:

WXML

JAVASCRIPT

WXSS

©2013-2025 Tencent Cloud International Pte. Ltd.

```
<view class="page-body">
  <view class="btn-area" id="buttonContainer">
    <button type="primary">Primary Page Action Normal</button>
    <button type="primary" loading="true">Primary Page Action Loading</button>
    <button type="primary" disabled="true">Primary Page Action
Disabled</button>

    <button type="default">Secondary Page Action Normal</button>
    <button type="default" disabled="true">Secondary Page Action
Disabled</button>

    <button type="warn">Warning Action Normal</button>
    <button type="warn" disabled="true">Warning Action Disabled</button>

    <view class="button-sp-area">
      <button type="primary" plain="true">Button</button>
      <button type="primary" disabled="true" plain="true">Disabled
button</button>

      <button type="default" plain="true">Button</button>
      <button type="default" disabled="true" plain="true">Button</button>

      <button class="mini-btn" type="primary" size="mini">Button</button>
      <button class="mini-btn" type="default" size="mini">Button</button>
      <button class="mini-btn" type="warn" size="mini">Button</button>
    </view>
  </view>
</view>
```

```
const types = ['default', 'primary', 'warn']
const pageObject = {
  data: {
    defaultSize: 'default',
    primarySize: 'default',
    warnSize: 'default',
    disabled: false,
    plain: false,
    loading: false
  },

  onShareAppMessage() {
    return {
      title: 'button',
      path: 'page/component/pages/button/button'
    }
  },
```

```
  setDisabled() {
    this.setData({
      disabled: !this.data.disabled
    })
  },

  setPlain() {
    this.setData({
      plain: !this.data.plain
    })
  },

  setLoading() {
    this.setData({
      loading: !this.data.loading
    })
  },

  handleContact(e) {
    console.log(e.detail)
  },

  handleGetPhoneNumber(e) {
    console.log(e.detail)
  },

  handleGetUserInfo(e) {
    console.log(e.detail)
  },

  handleOpenSetting(e) {
    console.log(e.detail.authSetting)
  },

  handleGetUserInfo(e) {
    console.log(e.detail.userInfo)
  }
}

for (let i = 0; i < types.length; ++i) {
  (function (type) {
    pageObject[type] = function () {
      const key = type + 'Size'
      const changedData = {}
      changedData[key] =
        this.data[key] === 'default' ? 'mini' : 'default'
```

```
        this.setData(changedData)
      }
    }(types[i]))
}

Page(pageObject)
```

```
button{
  margin-top: 30rpx;
  margin-bottom: 30rpx;
}
.button-sp-area{
  margin: 0 auto;
  width: 60%;
}
.mini-btn{
  margin-right: 10rpx;
}
```

checkbox

Feature description: Checkbox.

Parameter and description:

Property	Type	Default value	Required	Description
value	string	-	False	Checkbox identifier. When selected, it triggers the change event of the checkbox-group and carries the checkbox's value.
disabled	boolean	false	False	Whether the checkbox is disabled.
checked	boolean	false	False	Whether the checkbox is currently selected. The checkbox can be set to selected by default.
color	string	#09BB07	False	Checkbox color, similar to the CSS color property.

Example:

WXML

JAVASCRIPT

```
<view class="container">
  <view class="page-body">
    <view class="page-section page-section-gap">
      <view class="page-section-title">Default style</view>
      <label class="checkbox">
        <checkbox value="cb" checked="true"/>Selected
      </label>
      <label class="checkbox">
        <checkbox value="cb" />Unselected
      </label>
    </view>

    <view class="page-section">
      <view class="page-section-title">Recommended display style</view>
      <view class="weui-cells weui-cells_after-title">
        <checkbox-group bindchange="checkboxChange">
          <label class="weui-cell weui-check__label" wx:for="{{items}}"
wx:key="{{item.value}}">
            <view class="weui-cell__hd">
              <checkbox value="{{item.value}}" checked="{{item.checked}}"/>
            </view>
            <view class="weui-cell__bd">{{item.name}}</view>
          </label>
        </checkbox-group>
      </view>
    </view>
  </view>

</view>
```

```
Page({
  onShareAppMessage() {
    return {
      title: 'checkbox',
      path: 'page/component/pages/checkbox/checkbox'
    }
  },

  data: {
    items: [
      {value: 'USA', name: 'United States'},
      {value: 'CHN', name: 'China', checked: 'true'},
      {value: 'BRA', name: 'Brazil'},
      {value: 'JPN', name: 'Japan'},
      {value: 'ENG', name: 'United Kingdom'},
      {value: 'FRA', name: 'France'}
```

```
    ]
  },

  checkboxChange(e) {
    console.log('A change event is triggered in the checkbox, and the carried value

    const items = this.data.items
    const values = e.detail.value
    for (let i = 0, lenI = items.length; i < lenI; ++i) {
      items[i].checked = false

      for (let j = 0, lenJ = values.length; j < lenJ; ++j) {
        if (items[i].value === values[j]) {
          items[i].checked = true
          break
        }
      }
    }

    this.setData({
      items
    })
  }
})
```

checkbox-group

Feature description:A multiple selection component, which consists of multiple checkboxes.

Parameter and description:

Property	Type	Default value	Required	Description
bindchange	eventhandle	-	False	A change event is triggered when the selected checkboxes in the checkbox-group change, detail = {value:[array of values of selected checkboxes]}.

editor

Feature description:A rich text editor that allows editing of images and text.

The editor can export content as `HTML` and `plain text`. Internally, it uses the `delta` format for storage.

When setting content via the `setContents` API, parsing inserted `HTML` may result in errors due to illegal tags. It is recommended to use `delta` for insertion within the app.

The rich text component includes basic styles to ensure proper content display, which can be overridden during development.

The configurations to the image controls are only effective if set during initialization.

Related API:EditorContext.

Parameter and description:

Property	Type	Default value	Required	Description
read-only	boolean	false	False	Specifies whether the editor is set to read-only.
placeholder	string	-	False	Placehoder text.
show-img-size	boolean	false	False	Displays image size controls when an image is tapped.
show-img-toolbar	boolean	false	False	Displays toolbar controls when an image is tapped.
show-img-resize	boolean	false	False	Displays resize controls when an image is tapped.
bindready	eventhandle	-	False	Triggered when the editor is initialized.
bindfocus	eventhandle	-	False	Triggered when the editor gains focus. event.detail = {html, text, delta}.
bindblur	eventhandle	-	False	Triggered when the editor loses focus, detail = {html, text, delta}.
bindinput	eventhandle	-	False	Triggered when the editor content changes. detail = {html, text, delta}.
bindstatuschange	eventhandle	-	False	Triggered when the editor's internal style is changed via Context methods. Returns the styles set in the selection.

The editor supports certain `HTML` tags and inline styles but does not support `class` and `id attributes.`

Supported tags

Unsupported tags will be ignored or converted to supported tags (e.g., `<div>` will be converted to `<p>` supported tags.

Type	Node
Inline elements	`<span>` `<strong>` `<b>` `<ins>` `<em>` `<i>` `<u>` `<a>` `<del>` `<s>` `<sub>` `<sup>` `<img>`
Block-level elements	`<p>` `<h1>` `<h2>` `<h3>` `<h4>` `<h5>` `<h6>` `<hr>` `<ol>` `<ul>` `<li>`

Supported inline styles

Inline styles can only be applied to inline elements or block elements, but not both simultaneously. For example, `font-size` is categorized as an inline element property, so setting it on a `p` tag is invalid.

Type	Style
Block-level styles	`text-align` , `direction` , `margin` , `margin-top` , `margin-left` , `margin-right` , `margin-bottom` `padding` , `padding-top` , `padding-left` , `padding-right` , `padding-bottom` , `line-height` , `text-indent`
Inline styles	`font` , `font-size` , `font-style` , `font-variant` , `font-weight` , `font-family` , `letter-spacing` , `text-decoration` , `color` , `background-color`

Notes:

Event bindings in the inserted HTML will be removed.

When pasting, only the plain text content will be copied into the editor.

When inserting HTML into the editor, some unnecessary tags will be deleted to ensure the consistency of the content. For example, xxx will be rewritten as xxx .

The page will be pushed up when the editor gains focus to ensure the editing area is visible, which is a system behavior.

form

Feature description:A form, which submits user input from components like slider,picker ,checkbox,radio,switch and input .

When a user tap the button component where the form-type is set to submit in the form, the values of the form component will be submitted. Each form component should have a name attribute to serve as the key.

Parameter and description:

Property	Type	Default value	Required	Description

bindsubmit	eventhandle	-	False	A submit event is triggered when the form is submitted, carrying the data from the form. event.detail = {value : {'name': 'value'} , formId: ''}.
bindreset	eventhandle	-	False	A reset event is triggered when the form is reset.

Example:

WXML

JAVASCRIPT

```
<view class="container">
  <view class="page-body">
    <form catchsubmit="formSubmit" catchreset="formReset">
      <view class="page-section page-section-gap">
        <view class="page-section-title">switch</view>
        <switch name="switch"/>
      </view>

      <view class="page-section page-section-gap">
        <view class="page-section-title">radio</view>
        <radio-group name="radio">
          <label><radio value="radio1"/>Option 1</label>
          <label><radio value="radio2"/>Option 2</label>
        </radio-group>
      </view>

      <view class="page-section page-section-gap">
        <view class="page-section-title">checkbox</view>
        <checkbox-group name="checkbox">
          <label><checkbox value="checkbox1"/>Option 1</label>
          <label><checkbox value="checkbox2"/>Option 2</label>
        </checkbox-group>
      </view>

      <view class="page-section page-section-gap">
        <view class="page-section-title">slider</view>
        <slider value="50" name="slider" show-value ></slider>
      </view>

      <view class="page-section">
        <view class="page-section-title">input</view>
        <view class="weui-cells weui-cells_after-title">
          <view class="weui-cell weui-cell_input">
            <view class="weui-cell__bd" style="margin: 30rpx 0" >
```

```
              <input class="weui-input" name="input" placeholder="This is an
input box" />
            </view>
          </view>
        </view>
      </view>

      <view class="btn-area">
        <button style="margin: 30rpx 0" type="primary"
formType="submit">Submit</button>
        <button style="margin: 30rpx 0" formType="reset">Reset</button>
      </view>
    </form>
  </view>
</view>
```

```
Page({
  onShareAppMessage() {
    return {
      title: 'form',
      path: 'page/component/pages/form/form'
    }
  },

  data: {
    pickerHidden: true,
    chosen: ''
  },

  pickerConfirm(e) {
    this.setData({
      pickerHidden: true
    })
    this.setData({
      chosen: e.detail.value
    })
  },

  pickerCancel() {
    this.setData({
      pickerHidden: true
    })
  },

  pickerShow() {
    this.setData({
```

```
      pickerHidden: false
    })
  },

  formSubmit(e) {
    console.log('A submit event is triggered in the form, and the carried data is',
  },

  formReset(e) {
    console.log('A reset event is triggered in the form, and the carried data is',
    this.setData({
      chosen: ''
    })
  }
})
```

input

Feature description: An input field. This component is a Native Component and there are some restrictions for using it.

Parameter and description:

Property	Type	Valid values and description	Default value	Required	Description
value	string	-	-	True	Initial content of the input field.
type	string	text: Text input keyboard number: Numeric input keyboard idcard: ID number input keyboard digit: Numeric keyboard	text	False	Input type.

		with a decimal point nickname (Only supported on SaaS): Nickname input keyboard			
password	boolean	-	false	False	Whether the input is of password type.
placeholder	string	-		True	Placeholder text when the input field is empty.
placeholder-style	string	-	-	True	Specifies the placeholder style.
placeholder-class	string	-	input-placeholder	False	Specifies the placeholder style class.
disabled	boolean	-	false	False	Whether the input field is disabled.
maxlength	number	-	140	False	Maximum input length. Set to -1 for no limit.
cursor-spacing	number	-	0	False	Distance between the cursor and the keyboard. The minimum value between the input's bottom distance and cursor-spacing is used.
auto-focus	boolean	-	false	False	(Deprecated, use focus instead)

					Automatically focuses and brings up the keyboard
focus	boolean	-	false	False	Specifies whether to get the focus.
confirm-type	string	send: Sets the button in the bottom-right corner to "Send" search: Sets the button in the bottom-right corner to "Search" next: Sets the button in the bottom-right corner to "Next" go: Sets the button in the bottom-right corner to "Go" done: Sets the button in the bottom-right corner to "Done"	done	False	Sets the text of the button in the bottom-right corner of the keyboard. Effective only when type='text'.
confirm-hold	boolean	-	false	False	Whether to keep the keyboard open when the button in the bottom-right corner is tapped.
cursor	number	-	-	True	Specifies the position of the cursor when the

					input field gets focus.
selection-start	number	-	-1	False	Start position of the selection, effective when auto-focusing, should be used with selection-end.
selection-end	number	-	-1	False	End position of the selection, effective when auto-focusing, should be used with selection-start.
adjust-position	boolean	-	true	False	Whether to automatically adjust the page position when the keyboard pops up.
hold-keyboard	boolean	-	false	False	Whether to keep the keyboard open when tapping on the page.
bindinput	eventhandle	-	-	True	Triggered when the keyboard input occurs. event.detail = {value, cursor, keyCode}. The handler can return a string to replace the input field content.
bindfocus	eventhandle	-	-	True	Triggered when the input field gets focus.

					event.detail = { value, height }, where height is the keyboard height.
bindblur	eventhandle	-	-	True	Triggered when the input box loses focus, event.detail = { value, encryptedValue, encryptError }.
bindconfirm	eventhandle	-	-	True	Triggered when the user taps the Done button, event.detail = { value }.
bindkeyboardheightchange	eventhandle	-	-	True	Triggered when the keyboard height changes, event.detail = {height: height, duration: duration}.

Notes:

The final behavior of confirm-type depends on the implementation of the mobile input method itself. Some Android system input methods and third-party input methods may not support or fully support it.

The input component is a native component, and the font is the system font, so font-family cannot be set.

Avoid using CSS animations during the input focus period.

If the input is wrapped in a custom component and the form is outside of this custom component, the form will not be able to retrieve the value of the input within the custom component.

When the keyboard height changes, the keyboardheightchange event may be triggered multiple times. Developers should ignore events with the same height value to avoid redundant processing.

Example:

WXML

JAVASCRIPT

```
<view class="page-body">
  <view class="page-section">
```

```
    <view class="weui-cells__title">Input that can be auto-focused</view>
    <view class="weui-cells weui-cells_after-title">
      <view class="weui-cell weui-cell_input">
        <input class="weui-input" auto-focus placeholder="Will be focused"/>
      </view>
    </view>
  </view>
  <view class="page-section">
    <view class="weui-cells__title">Input that controls the max input
length</view>
    <view class="weui-cells weui-cells_after-title">
      <view class="weui-cell weui-cell_input">
        <input class="weui-input" maxlength="10" placeholder="The max input
length is 10" />
      </view>
    </view>
  </view>
  <view class="page-section">
    <view class="weui-cells__title">Get the input value in real-time:
{{inputValue}}</view>
    <view class="weui-cells weui-cells_after-title">
      <view class="weui-cell weui-cell_input">
        <input class="weui-input"  maxlength="10" bindinput="bindKeyInput"
placeholder="Input is synced to the view"/>
      </view>
    </view>
  </view>
  <view class="page-section">
    <view class="weui-cells__title">Input that the controls the typing</view>
    <view class="weui-cells weui-cells_after-title">
      <view class="weui-cell weui-cell_input">
        <input class="weui-input"  bindinput="bindReplaceInput"
placeholder="Typing two consecutive ones will generate a two" />
      </view>
    </view>
  </view>
  <view class="page-section">
    <view class="weui-cells__title">Input that controls the keyboard</view>
    <view class="weui-cells weui-cells_after-title">
      <view class="weui-cell weui-cell_input">
        <input class="weui-input"  bindinput="bindHideKeyboard"
placeholder="Typing 123 will automatically close the keyboard" />
      </view>
    </view>
  </view>
  <view class="page-section">
    <view class="weui-cells__title">Input for entering numbers</view>
```

```
    <view class="weui-cells weui-cells_after-title">
      <view class="weui-cell weui-cell_input">
        <input class="weui-input" type="number" placeholder="This is a number
input box" />
      </view>
    </view>
  </view>
  <view class="page-section">
    <view class="weui-cells__title">Input for entering a password</view>
    <view class="weui-cells weui-cells_after-title">
      <view class="weui-cell weui-cell_input">
        <input class="weui-input" password type="text" placeholder="This is a
password input box" />
      </view>
    </view>
  </view>
  <view class="page-section">
    <view class="weui-cells__title">Input with a decimal point</view>
    <view class="weui-cells weui-cells_after-title">
      <view class="weui-cell weui-cell_input">
        <input class="weui-input" type="digit" placeholder="Numeric keyboard
with a decimal point"/>
      </view>
    </view>
  </view>
  <view class="page-section">
    <view class="weui-cells__title">Input for entering ID card numbers</view>
    <view class="weui-cells weui-cells_after-title">
      <view class="weui-cell weui-cell_input">
        <input class="weui-input" type="idcard" placeholder="ID card input
keyboard" />
      </view>
    </view>
  </view>
  <view class="page-section">
    <view class="weui-cells__title">Input that controls the placeholder
color</view>
    <view class="weui-cells weui-cells_after-title">
      <view class="weui-cell weui-cell_input">
        <input class="weui-input" placeholder-style="color:#F76260"
placeholder="The color of the placeholder font is red" />
      </view>
    </view>
  </view>
</view>
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
Page({
  data: {
    focus: false,
    inputValue: ''
  },
  bindKeyInput: function (e) {
    this.setData({
      inputValue: e.detail.value
    })
  },
  bindReplaceInput: function (e) {
    var value = e.detail.value
    var pos = e.detail.cursor
    var left
    if (pos !== -1) {
      // The cursor is in the middle
      left = e.detail.value.slice(0, pos)
      // Calculates the cursor position
      pos = left.replace(/11/g, '2').length
    }

    // Returns the object directly, filters the input, and controls the cursor posi
    return {
      value: value.replace(/11/g, '2'),
      cursor: pos
    }

    // Or returns the string directly, with the cursor at the end
    // return value.replace(/11/g,'2'),
  },
  bindHideKeyboard: function (e) {
    if (e.detail.value === '123') {
      // Closes the keyboard
      wx.hideKeyboard()
    }
  }
})
```

label

Feature description: This component is used to improve the usability of form components.

Use the for property to find the corresponding id, or place the control under the label. When tapping, the corresponding control will be triggered. The for property has a higher priority than internal controls. When there are multiple internal controls, the first control is triggered by default. Currently, the following controls can be bound:button,checkbox,radio,switch and input.

Parameter and description:

Property	Type	Default value	Required	Description
for	string	-	False	The ID of the bound control.

Example:

WXML

JAVASCRIPT

WXSS

```
<view class="container">
  <view class="page-body">
    <view class="page-section page-section-gap">
      <view class="page-section-title">The form component is inside the
label</view>
      <checkbox-group class="group" bindchange="checkboxChange">
        <view class="label-1" wx:for="{{checkboxItems}}">
          <label>
            <checkbox value="{{item.name}}" checked="{{item.checked}}">
</checkbox>
            <text class="label-1-text">{{item.value}}</text>
          </label>
        </view>
      </checkbox-group>
    </view>

    <view class="page-section page-section-gap">
      <view class="page-section-title">Label uses for to identify the form
component</view>
      <radio-group class="group" bindchange="radioChange">
        <view class="label-2" wx:for="{{radioItems}}">
          <radio id="{{item.name}}" value="{{item.name}}" checked="
{{item.checked}}"></radio>
          <label class="label-2-text" for="{{item.name}}"><text>{{item.name}}
</text></label>
        </view>
      </radio-group>
    </view>

    <view class="page-section page-section-gap">
```

```
      <view class="page-section-title">If there are multiple controls in the
label, the first one is selected</view>
      <label class="label-3">
        <checkbox class="checkbox-3">Option 1</checkbox>
        <checkbox class="checkbox-3">Option 2</checkbox>
        <view class="label-3-text">When tapping the text under this label, the
first checkbox is selected by default</view>
      </label>
    </view>
  </view>
</view>
```

```
Page({
  onShareAppMessage() {
    return {
      title: 'label',
      path: 'page/component/pages/label/label'
    }
  },

  data: {
    checkboxItems: [
      {name: 'USA', value: 'United States'},
      {name: 'CHN', value: 'China', checked: 'true'}
    ],
    radioItems: [
      {name: 'USA', value: 'United States'},
      {name: 'CHN', value: 'China', checked: 'true'}
    ],
    hidden: false
  },

  checkboxChange(e) {
    const checked = e.detail.value
    const changed = {}
    for (let i = 0; i < this.data.checkboxItems.length; i++) {
      if (checked.indexOf(this.data.checkboxItems[i].name) !== -1) {
        changed['checkboxItems[' + i + '].checked'] = true
      } else {
        changed['checkboxItems[' + i + '].checked'] = false
      }
    }
    this.setData(changed)
  },

  radioChange(e) {
```

```
    const checked = e.detail.value
    const changed = {}
    for (let i = 0; i < this.data.radioItems.length; i++) {
      if (checked.indexOf(this.data.radioItems[i].name) !== -1) {
        changed['radioItems[' + i + '].checked'] = true
      } else {
        changed['radioItems[' + i + '].checked'] = false
      }
    }
    this.setData(changed)
  },

  tapEvent() {
    console.log('the button is tapped')
  }
})
```

```
.label-1, .label-2{
  margin: 30rpx 0;
}
.label-3-text{
  color: #576B95;
  font-size: 28rpx;
}
.checkbox-3{
  display: block;
  margin: 30rpx 0;
}
```

picker

Feature description:A scroll picker that pops up from the bottom.

Parameter and description:

Property	Type	Valid values and description	Default value	Required	Description
mode	string	selector: Normal picker multiSelector: Multi-column picker time: Time picker date: Date picker region: Region picker	selector	False	Type of the picker.

disabled	boolean	-	false	False	Whether the selector is disabled.
bindcancel	eventhandle	-	-	False	Triggered when selection is canceled.

In addition to the common properties above, the `picker` has different properties for different modes.

Normal picker: mode = selector

Property	Type	Default value	Description
range	array/object array	[]	Effective when mode is selector or multiSelector.
range-key	string	-	When range is an object array, specifies the key in the Object Array to display as the content of the picker.
value	number	0	Specifies which one in the range is selected (index starts from 0).
bindchange	eventhandle	-	The change event is triggered when value changes, event.detail = {value}.

Multi-column picker: mode = multiSelector

Property	Type	Default value	Description
range	array/object array	[]	Effective when mode is selector or multiSelector.
range-key	string	-	When range is an object array, specifies the key in the Object Array to display as the content of the picker.
value	array	[]	Specifies which one in the range is selected (index starts from 0).
bindchange	eventhandle	-	The change event is triggered when value changes, event.detail = {value}.
bindcolumnchange	eventhandle	-	Triggered when the column changes.

Time picker: mode = time

Property	Type	Default value	Description
value	string	-	Indicates the selected time, format is "hh:mm".
start	string	-	Indicates the start of the valid time range, format is "hh:mm".
end	string	-	Indicates the end of the valid time range, format is "hh:mm".
bindchange	eventhandle	-	The change event is triggered when value changes, event.detail = {value}.

Date picker: mode = date

Property	Type	Valid values	Default value	Description
value	string	-	Current day	Specifies the selected date in YYYY-MM-DD format.
start	string	-	-	Indicates the selected date, format is "YYYY-MM-DD".
end	string	-	-	Indicates the start of the valid date range, format is "YYYY-MM-DD".
fields	string	year: Year month: Month day: Day	day	Indicates the granularity of the picker.
bindchange	eventhandle	-	-	The change event is triggered when value changes, event.detail = {value}.

Region picker: mode = region

Property	Type	Default value	Description
value	Array	[]	Specifies the selected regions. The first value in each column is selected by default.
custom-item	string	-	Adds a custom item at the top of each column.
bindchange	evenhandle	-	The change event is triggered when value changes, event.detail = {value: value, code: code, postcode: postcode},

			where the code field is the statistical division code and postcode is the postal code.

Example:

WXML

JAVASCRIPT

```
<view class="section">
  <view class="section__title">Normal picker</view>
  <picker bindchange="bindPickerChange" value="{{index}}" range="{{array}}">
    <view class="picker">
      Currently selected: {{array[index]}}
    </view>
  </picker>
</view>
<view class="section">
  <view class="section__title">Multi-column picker</view>
  <picker mode="multiSelector" bindchange="bindMultiPickerChange"
bindcolumnchange="bindMultiPickerColumnChange" value="{{multiIndex}}" range="
{{multiArray}}">
    <view class="picker">
      Currently selected: {{multiArray[0][multiIndex[0]]}}, {{multiArray[1]
[multiIndex[1]]}}, {{multiArray[2][multiIndex[2]]}}
    </view>
  </picker>
</view>
<view class="section">
  <view class="section__title">Time picker</view>
  <picker mode="time" value="{{time}}" start="09:01" end="21:01"
bindchange="bindTimeChange">
    <view class="picker">
      Currently selected: {{time}}
    </view>
  </picker>
</view>

<view class="section">
  <view class="section__title">Date picker</view>
  <picker mode="date" value="{{date}}" start="2015-09-01" end="2017-09-01"
bindchange="bindDateChange">
    <view class="picker">
      Currently selected: {{date}}
    </view>
  </picker>
</view>
<view class="section">
```

```
  <view class="section__title">Region picker</view>
  <picker mode="region" bindchange="bindRegionChange" value="{{region}}"
custom-item="{{customItem}}">
    <view class="picker">
      Currently selected: {{region[0]}}, {{region[1]}}, {{region[2]}}
    </view>
  </picker>
</view>
```

```
Page({
  data: {
    array: ['United States', 'China', 'Brazil', 'Japan'],
    objectArray: [
      {
        id: 0,
        name: 'United States'
      },
      {
        id: 1,
        name: 'China'
      },
      {
        id: 2,
        name: 'Brazil'
      },
      {
        id: 3,
        name: 'Japan'
      }
    ],
    index: 0,
    multiArray: [['Invertebrates', 'Vertebrates'], ['Flatworms', 'Nematodes', 'Anne
    objectMultiArray: [
      [
        {
          id: 0,
          name: 'Invertebrates'
        },
        {
          id: 1,
          name: 'Vertebrates'
        }
      ], [
        {
          id: 0,
          name: 'Flatworms'
```

```
        },
        {
          id: 1,
          name: 'Nematodes'
        },
        {
          id: 2,
          name: 'Annelates'
        },
        {
          id: 3,
          name: 'Mollusks'
        },
        {
          id: 4,
          name: 'Arthropods'
        }
      ], [
        {
          id: 0,
          name: 'Pork tapeworms'
        },
        {
          id: 1,
          name: 'Blood-sucking worms'
        }
      ]
    ],
    multiIndex: [0, 0, 0],
    date: '2016-09-01',
    time: '12:01',
    region: ['Guangdong Province', 'Guangzhou', 'Haizhu District'],
    customItem: 'All'
  },
  bindPickerChange: function(e) {
    console.log('The picker sends selection change, and the carried value is' e.det
    this.setData({
      index: e.detail.value
    })
  },
  bindMultiPickerChange: function (e) {
    console.log('The picker sends selection change, and the carried value is' e.det
    this.setData({
      multiIndex: e.detail.value
    })
  },
  bindMultiPickerColumnChange: function (e) {
```

```
console.log('The modified column is', e.detail.column, ', and the value is', e.
var data = {
  multiArray: this.data.multiArray,
  multiIndex: this.data.multiIndex
};
data.multiIndex[e.detail.column] = e.detail.value;
switch (e.detail.column) {
  case 0:
    switch (data.multiIndex[0]) {
      case 0:
        data.multiArray[1] = ['Flatworms', 'Nematodes', 'Annelates', 'Mollusks'
        data.multiArray[2] = ['Pork tapeworms', 'Blood-sucking worms'];
        break;
      case 1:
        data.multiArray[1] = ['Fish', ''Amphibians', 'Reptiles'];
        data.multiArray[2] = ['Carassius auratus', 'Hairtail'];
        break;
    }
    data.multiIndex[1] = 0;
    data.multiIndex[2] = 0;
    break;
  case 1:
    switch (data.multiIndex[0]) {
      case 0:
        switch (data.multiIndex[1]) {
          case 0:
            data.multiArray[2] = ['Pork tapeworms', 'Blood-sucking worms'];
            break;
          case 1:
            data.multiArray[2] = ['Roundworms'];
            break;
          case 2:
            data.multiArray[2] = ['Ants', 'Leech'];
            break;
          case 3:
            data.multiArray[2] = ['Mussels', 'Snails', 'Slugs'];
            break;
          case 4:
            data.multiArray[2] = ['Insects', 'Crustaceans', 'Arachnids', 'Myria
            break;
        }
        break;
      case 1:
        switch (data.multiIndex[1]) {
          case 0:
            data.multiArray[2] = ['Carassius auratus', 'Hairtail'];
            break;
```

```
                case 1:
                  data.multiArray[2] = [''Frogs', 'Giant salamanders'];
                  break;
                case 2:
                  data.multiArray[2] = ['Lizards', 'Turtles', 'Geckos'];
                  break;
              }
              break;
          }
          data.multiIndex[2] = 0;
          break;
      }
      console.log(data.multiIndex);
      this.setData(data);
    },
    bindDateChange: function(e) {
      console.log('The picker sends selection change, and the carried value is' e.det
      this.setData({
        date: e.detail.value
      })
    },
    bindTimeChange: function(e) {
      console.log('The picker sends selection change, and the carried value is' e.det
      this.setData({
        time: e.detail.value
      })
    },
    bindRegionChange: function (e) {
      console.log('The picker sends selection change, and the carried value is' e.det
      this.setData({
        region: e.detail.value
      })
    }
  })
```

picker-view

Feature description: An embedded scroll picker for the page. Only picker-view-column component can be placed, and other nodes will not be displayed.

Parameter and description:

Property	Type	Default value	Required	Description

value	Array.<number>	-	False	The numbers in the array represent the selected item in each picker-view-column (index starts from 0). If the number exceeds the length of the options, the last item is selected.
mask-class	string	-	False	Sets the class name for the mask layer.
indicator-style	string	-	False	Sets the style of the selected box in the picker.
bindchange	eventhandle	-	False	The change event is triggered when scrolling selection changes, event.detail = {value}; value is an array indicating the current selection in each picker-view-column (index starts from 0).
bindpickstart	eventhandle	-	False	The event is triggered when scrolling selection starts.
bindpickend	eventhandle	-	False	The event is triggered when scrolling selection ends.
indicator-class	string	-	False	Sets the class name of the selected box in the picker.
mask-style	string	-	False	Sets the style of the mask layer.
immediate-change	boolean	false	False	Whether to trigger the change event immediately when the finger is released. If not enabled, the change event will be triggered after the scrolling animation ends.

Example:

WXML

JAVASCRIPT

```
<view class="container">
  <view class="page-body">
    <view class="selected-date">{{year}}Year{{month}}Month{{day}}Day{{isDaytime
? "Day" : "Night"}}</view>
    <picker-view indicator-style="height: 50px;" style="width: 100%; height:
300px;" value="{{value}}" bindchange="bindChange">
      <picker-view-column>
```

```
        <view wx:for="{{years}}" wx:key="{{years}}" style="line-height: 50px;
text-align: center;">{{item}}Year</view>
      </picker-view-column>
      <picker-view-column>
        <view wx:for="{{months}}" wx:key="{{months}}" style="line-height: 50px;
text-align: center;">{{item}}Month</view>
      </picker-view-column>
      <picker-view-column>
        <view wx:for="{{days}}" wx:key="{{days}}" style="line-height: 50px;
text-align: center;">{{item}}Day</view>
      </picker-view-column>
      <picker-view-column>
        <view class="icon-container">
          <image class="picker-icon" src="../lib/daytime.png" />
        </view>
        <view class="icon-container">
          <image class="picker-icon" src="../lib/night.png" />
        </view>
      </picker-view-column>
    </picker-view>
  </view>

</view>
```

```
const date = new Date()
const years = []
const months = []
const days = []

for (let i = 1990; i <= date.getFullYear(); i++) {
  years.push(i)
}

for (let i = 1; i <= 12; i++) {
  months.push(i)
}

for (let i = 1; i <= 31; i++) {
  days.push(i)
}

Page({
  onShareAppMessage() {
    return {
      title: 'picker-view',
      path: 'page/component/pages/picker-view/picker-view'
```

```
    }
  },

  data: {
    years,
    year: date.getFullYear(),
    months,
    month: 2,
    days,
    day: 2,
    value: [9999, 1, 1],
    isDaytime: true,
  },

  bindChange(e) {
    const val = e.detail.value
    this.setData({
      year: this.data.years[val[0]],
      month: this.data.months[val[1]],
      day: this.data.days[val[2]],
      isDaytime: !val[3]
    })
  }
})
```

picker-view-column

Feature description:A sub-item of the scroll picker. This component can only be placed in picker-view, and the height of its child nodes will be automatically match the height of the selection box in picker-view.

radio

Feature description:A single selection item.

Parameter and description

Property	Type	Default value	Required	Description
value	string	-	False	Identifier of the radio. When the radio is selected, the change event of radio-group is triggered and the radio value will be carried.

checked	boolean	false	False	Whether the radio is selected.
disabled	boolean	false	False	Whether the radio is disabled.
color	string	#09BB07	False	The color of the radio, similar to the CSS color property.

Example:

WXML

JAVASCRIPT

```
<view class="page-body">
  <view class="page-section">
    <view class="page-section-title">Default style</view>
    <label class="radio">
      <radio value="r1" checked="true"/>Selected
    </label>
    <label class="radio">
      <radio value="r2" />Unselected
    </label>
  </view>


  <view class="page-section">
    <view class="page-section-title">Recommended display style</view>
    <view class="weui-cells weui-cells_after-title">
      <radio-group bindchange="radioChange">
        <label class="weui-cell weui-check__label" wx:for="{{items}}" wx:key="
{{item.value}}">

          <view class="weui-cell__hd">
            <radio value="{{item.value}}" checked="true"/>
          </view>
          <view class="weui-cell__bd">{{item.name}}</view>
        </label>
      </radio-group>
    </view>
  </view>
</view>
```

```
Page({
  onShareAppMessage() {
    return {
      title: 'radio',
      path: 'page/component/pages/radio/radio'
```

```
    }
  },

  data: {
    items: [
      {value: 'USA', name: 'United States'},
      {value: 'CHN', name: 'China', checked: 'true'},
      {value: 'BRA', name: 'Brazil'},
      {value: 'JPN', name: 'Japan'},
      {value: 'ENG', name: 'United Kingdom'},
      {value: 'FRA', name: 'France'},
    ]
  },

  radioChange(e) {
    console.log('A change event is triggered in the radio, and the carried value is

    const items = this.data.items
    for (let i = 0, len = items.length; i < len; ++i) {
      items[i].checked = items[i].value === e.detail.value
    }

    this.setData({
      items
    })
  }
})
```

radio-group

Feature description:A single selection group composed of multiple radio items.

Parameter and description

Property	Type	Default value	Required	Description
bindchange	eventhandle	-	False	A change event is triggered when the selected item in the radio-group component changes, detail = {value: [array of values of selected radio buttons]}.

slider

Feature description:A sliding picker.

Parameter and description

Property	Type	Default value	Required	Description
min	number	0	False	Minimum value.
max	number	100	False	Maximum value.
step	number	1	False	Step size, must be greater than 0 and divisible by (max - min).
disabled	boolean	false	False	Whether the slider is disabled.
value	number	0	False	Current value.
color	color	#e9e9e9	False	Background bar color (use backgroundColor).
selected-color	color	#1aad19	False	Selected color (use activeColor).
activeColor	color	#1aad19	False	Selected color.
backgroundColor	color	#e9e9e9	False	Background bar color.
block-size	number	28	False	Size of the slider block, range is 12 - 28.
block-color	color	#ffffff	False	Color of the slider block.
show-value	boolean	false	False	Whether to display the current value.
bindchange	eventhandle	-	False	Event triggered after a drag is completed, event.detail = {value}.
bindchanging	eventhandle	-	False	Event triggered during the drag, event.detail = {value}.

Example:

WXML

JAVASCRIPT

```
<view class="page">
    <view class="page__hd">
        <text class="page__title">slider</text>
```

```
        <text class="page__desc">Slider</text>
    </view>
    <view class="page__bd">
        <view class="page-section page-section-gap">
          <view class="page-section-title">Set the step</view>
          <view class="body-view">
            <slider value="60" bindchange="slider2change" step="5"/>
          </view>
        </view>

        <view class="page-section page-section-gap">
          <view class="page-section-title">Show the current value</view>
          <view class="body-view">
            <slider value="50" bindchange="slider3change" show-value/>
          </view>
        </view>

        <view class="page-section page-section-gap">
          <view class="page-section-title">Set the min/max values</view>
          <view class="body-view">
            <slider value="100" bindchange="slider4change" min="50" max="200"
show-value/>
          </view>
        </view>
    </view>
</view>
```

```
var pageData = {}
for (var i = 1; i < 5; ++i) {
  (function (index) {
    pageData[`slider${index}change`] = function (e) {
      console.log(`A change event is triggered in slider${index}, and the carried v
    }
  })(i);
}
Page(pageData)
```

switch

Feature description:A switch picker.

Parameter and description

Property	Type	Default value	Required	Description

checked	boolean	false	False	Whether the switch is selected.
disabled	boolean	false	False	Whether the switch is disabled.
type	string	switch	False	Style of the switch, valid values: switch, checkbox.
color	string	#04BE02	False	Color of the switch, similar to the CSS color property.
bindchange	eventhandle	-	False	The change event is triggered when the component is tapped and the value of checked changes, event.detail={ value}.

Example:

WXML

JAVASCRIPT

```
<view class="page">
    <view class="page__hd">
        <text class="page__title">switch</text>
        <text class="page__desc">Switch</text>
    </view>
    <view class="page__bd">
        <view class="section section_gap">
            <view class="section__title">type="switch"</view>
            <view class="body-view">
                <switch checked="{{switch1Checked}}"
bindchange="switch1Change"/>
            </view>
        </view>

        <view class="section section_gap">
            <view class="section__title">type="checkbox"</view>
            <view class="body-view">
                <switch type="checkbox" checked="{{switch2Checked}}"
bindchange="switch2Change"/>
            </view>
        </view>
    </view>
</view>
```

```
var pageData = {
  data: {
```

```
    switch1Checked: true,
    switch2Checked: false,
    switch1Style: '',
    switch2Style: 'text-decoration: line-through'
  }
}
for (var i = 1; i <= 2; ++i) {
  (function (index) {
    pageData[`switch${index}Change`] = function (e) {
      console.log(`A change event is triggered in switch${index}, and the carried v
      var obj = {}
      obj[`switch${index}Checked`] = e.detail.value
      this.setData(obj)
      obj = {}
      obj[`switch${index}Style`] = e.detail.value ? '' : 'text-decoration: line-thr
      this.setData(obj)
    }
  })(i)
}
Page(pageData)
```

textarea

Feature description:A multi-line input box. This component is a Native Component and there are some restrictions for using it.

Parameter and description:

Property	Type	Valid values and description	Default value	Required	Description
value	string	-	-	False	Content of the input box.
placeholder	string	-	-	False	Placeholder text when the input field is empty.
placeholder-style	string	-	-	False	Style for the placeholder, currently supports color, font-size, and font-weight.

placeholder-class	string	-	textarea-placeholder	False	Style class for the placeholder.
fixed	boolean	-	false	False	If the textarea is in a position:fixed area, the property fixed should be set to true.
show-confirm-bar	boolean	-	true	False	Whether to show the bar with the "Done" button above the keyboard.
disabled	boolean	-	false	False	Whether the textarea is disabled.
maxlength	number	-	140	False	Maximum input length. Set to -1 for no limit.
auto-focus	boolean	-	false	False	Automatically focuses and brings up the keyboard.
focus	boolean	-	false	False	Gets the focus.
auto-height	boolean	-	false	False	Whether to enable automatic height adjustment for the input box. If auto-height is set to true, the style.height is invalid.
cursor-spacing	number / string	-	0	False	Distance between the cursor and the keyboard. Use the smaller value between the distance from the textarea to the bottom and the distance specified

©2013-2025 Tencent Cloud International Pte. Ltd.

					by cursor-spacing as the distance between the cursor and the keyboard.
cursor	number	-	-1	False	Cursor position when focused.
selection-start	number	-	-1	False	Start position of the cursor, effective when auto-focusing, used with `selection-end` .
selection-end	number	-	-1	False	End position of the cursor, effective when auto-focusing, used with `selection-start` .
adjust-position	boolean	-	true	False	Automatically pushes the page up when the keyboard pops up.
hold-keyboard	boolean	-	false	False	Whether to keep the keyboard open when tapping on the page.
confirm-type	string	send: Sets the button in the bottom-right corner to "Send"; search: Sets the button in the bottom-right	return	False	Specifies the text on the button in the bottom-right corner of the keyboard.

		corner to "Search"; next: Sets the button in the bottom-right corner to "Next"; go: Sets the button in the bottom-right corner to "Go"; done: Sets the button in the bottom-right corner to "Done"; return: Sets the button in the bottom-right corner to “Carriage return”.			
confirm-hold	boolean	-	false	False	Whether to keep the keyboard open when the button in the bottom-right corner is tapped.
bindfocus	eventhandle	-	-	False	Triggered when the input field gets focus. event.detail = { value, height }, where height is the keyboard height.

bindblur	eventhandle	-	-	False	Triggered when the input box loses focus, event.detail = {value, cursor}.
bindlinechange	eventhandle	-	-	False	This event is triggered when the number of lines in the input box changes, event.detail = {height: 0, heightRpx: 0, lineCount: 0}
bindinput	eventhandle	-	-	False	When keyboard input occurs, the input event is triggered, with event.detail = {value, cursor, keyCode}. The keyCode represents the key value, but currently, the tool does not support returning the keyCode parameter. The return value of the bindinput handler will not be reflected in the textarea.
bindconfirm	eventhandle	-	-	False	Triggered when the user taps the Done button, event.detail = {value: value}.
bindkeyboardheightchange	eventhandle	-	-	False	Triggered when the keyboard height changes, event.detail = {height: height, duration: duration}

©2013-2025 Tencent Cloud International Pte. Ltd.

Notes:

The blur event in textarea will be later than the tap event on the page. If you need to get the textarea in the tap event of the button, use bindsubmit of the form component.

It is not recommended to modify user input on multi-line text, so the return value of the bindinput processing function will not be reflected on the textarea.

When the keyboard height changes, the keyboardheightchange event may be triggered multiple times. Developers should ignore events with the same height value to avoid redundant processing.

Example:

WXML

JAVASCRIPT

```
<view class="section">
  <textarea bindblur="bindTextAreaBlur" auto-height placeholder="Automatic
height adjustment" />
</view>
<view class="section">
  <textarea placeholder="The placeholder color is red" placeholder-
style="color:red;"  />
</view>
<view class="section">
  <textarea placeholder="This is a textarea that can be automatically focused"
auto-focus />
</view>
<view class="section">
  <textarea placeholder="This will only be focused when the button is tapped"
focus="{{focus}}" />
  <view class="btn-area">
    <button bindtap="bindButtonTap">Make the input box focused</button>
  </view>
</view>
<view class="section">
  <form bindsubmit="bindFormSubmit">
    <textarea placeholder="Textarea in the form component" name="textarea"/>
    <button form-type="submit"> Submit </button>
  </form>
</view>
```

```
Page({
  data: {
    height: 20,
    focus: false
  },
  bindButtonTap: function() {
    this.setData({
```

```
      focus: true
    })
  },
  bindTextAreaBlur: function(e) {
    console.log(e.detail.value)
  },
  bindFormSubmit: function(e) {
    console.log(e.detail.value.textarea)
  }
})
```

API

API Overview

Last updated：2025-06-09 10:08:05

Basic

System capabilities

Name	Feature description	Minimum version
canIUse	Checks whether the APIs, callbacks, parameters, and components of the mini program are supported in the current version.	1.5.0
env	Environment variables.	1.5.1
base64ToArrayBuffer	Converts a Base64 string into an ArrayBuffer object.	1.5.1
arrayBufferToBase64	Converts an ArrayBuffer object to a Base64 string.	1.5.1

System

Name	Feature description	Minimum version
getSystemInfo	Asynchronously gets system information.	1.5.0
getSystemInfoSync	Synchronously gets system information.	1.5.1
getSystemInfoAsync	Asynchronously gets system information.	1.5.1
getWindowInfo	Gets window information.	1.5.1
getSystemSetting	Gets device settings.	1.5.0
openSystemBluetoothSetting	Navigates to the system’s Bluetooth settings page.	1.5.1
openAppAuthorizeSetting	Navigates to the app's authorization management page.	1.5.1
getRendererUserAgent	Gets the userAgent of the WebView component that loads the mini program.	1.5.1
getDeviceInfo	Gets basic information of the device.	1.5.1

©2013-2025 Tencent Cloud International Pte. Ltd.

getAppBaseInfo	Gets basic information of the superapp.	1.5.1
getAppAuthorizeSetting	Gets authorization settings of the superapp.	1.5.1

Update

Name	Feature description	Minimum version
getUpdateManager	Gets the global unique version update manager to manage mini program updates.	1.5.1
UpdateManager.applyUpdate	Forces the mini program to restart and use the new version.	1.5.0
UpdateManager.onCheckForUpdate	Listens for the event of the update check request to the backend.	1.5.0
UpdateManager.onUpdateFailed	Listens for mini program update failure event.	1.5.0
UpdateManager.onUpdateReady	Listens for mini program update event.	1.5.0

Mini program

Name	Feature description	Minimum version
getLaunchOptionsSync	Gets the parameters when the mini program starts up.	1.5.0
getEnterOptionsSync	Gets the parameters when the mini program starts up.	1.5.1
onError	Listens for mini program error event.	1.5.0
offError	Removes the listener for mini program error event.	1.5.0
onThemeChange	Listens for system theme change event.	1.5.0
offThemeChange	Removes the listener for system theme change event.	1.5.0
onPageNotFound	Listens for the event when the mini program attempts to open a non-existent page.	1.5.1
offPageNotFound	Removes the listener for the event when the mini program attempts to open a non-existent page.	1.5.1
onAppShow	Listens for the event when the mini program switches to the frontend.	1.5.1

©2013-2025 Tencent Cloud International Pte. Ltd.

offAppShow	Removes the listener for the event when the mini program switches to the frontend.	1.5.1
onAppHide	Listens for the event when the mini program switches to the backend.	1.5.1
offAppHide	Removes the listener for the event when the mini program switches to the frontend.	1.5.1
onUnhandledRejection	Listens for unhandled Promise rejection event.	1.5.1
offUnhandledRejection	Removes the listener for unhandled Promise rejection event.	1.5.1

Debug

Name	Feature description	Minimum version
setEnableDebug	Sets whether to enable the debugging.	1.5.0
getLogManager	Gets the log manager object.	1.5.1
console.debug	Outputs debug logs to the debug panel.	1.5.0
console.error	Outputs error logs to the debug panel.	1.5.0
console.group	Creates a new group in the debug panel.	1.5.0
console.groupEnd	Ends a group created by console.group.	1.5.0
console.info	Outputs info logs to the debug panel.	1.5.0
console.log	Outputs log messages to the debug panel.	1.5.0
console.warn	Outputs warn logs to the debug panel.	1.5.0
LogManager.debug	Writes debug logs.	1.5.0
LogManager.info	Writes info logs.	1.5.0
LogManager.log	Writes log messages.	1.5.0
LogManager.warn	Writes warning logs.	1.5.0
RealtimeTagLogManager.addFilterMsg	Adds filter keywords.	1.5.16
RealtimeTagLogManager.error	Writes error logs.	1.5.16

©2013-2025 Tencent Cloud International Pte. Ltd.

RealtimeTagLogManager.info	Writes info logs.	1.5.16
RealtimeTagLogManager.setFilterMsg	Sets filter keywords.	1.5.16
RealtimeTagLogManager.warn	Writes warning logs.	1.5.16

Navigation between pages within a mini program

Name	Feature description	Minimum version
switchTab	Navigates to the tab bar page and closes all other non-tab bar pages.	1.5.0
reLaunch	Closes all pages and opens a specified page in the app.	1.5.0
redirectTo	Closes the current page and navigates to a specific page within the app.	1.5.0
navigateTo	Retains the current page and navigates to a specific page within the app.	1.5.1
navigateBack	Closes the current page and returns to the previous page or a multi-level page.	1.5.0
EventChannel	Inter-page event communication channel.	1.5.1

Navigation between different mini programs

Name	Feature description	Minimum version
exitMiniProgram	Exits the current mini program.	1.5.0
navigateToMiniProgram	Opens another mini program.	1.5.0
navigateBackMiniProgram	Returns to the previous mini program. This API can be called successfully only when the current mini program is opened by another mini program.	1.5.1

Share

©2013-2025 Tencent Cloud International Pte. Ltd.

Name	Feature description	Minimum version
hideShareMenu	Hides the Share button on the current page.	1.5.0
showShareMenu	Shows the Share button on the current page.	1.5.0
updateShareMenu	Updates the sharing properties.	1.5.0
showShareImageMenu	Opens the pop-up window for image sharing.	1.5.35
onCopyUrl	Listens for the event triggered when the user taps the "Copy URL" button in the top-right menu.	1.5.35
offCopyUrl	Removes all listeners for event triggered when the user taps the "Copy URL" button in the top-right menu.	1.5.35

User interface

Interaction

Name	Feature description	Minimum version
hideLoading	Hides the loading prompt.	1.5.0
hideToast	Hides the message prompt.	1.5.0
showActionSheet	Shows the action menu.	1.5.0
showLoading	Shows the loading prompt.	1.5.0
showModal	Shows the modal dialog.	1.5.0
showToast	Shows the message prompt.	1.5.0
enableAlertBeforeUnload	Enables a confirmation dialog when the user tries to navigate away from a page in the mini program.	1.5.1
disableAlertBeforeUnload	Disables the confirmation dialog when the user tries to navigate away from a page in the mini program.	1.5.1

Navigation bar

Name	Feature description	Minimum

©2013-2025 Tencent Cloud International Pte. Ltd.

		version
setNavigationBarTitle	Dynamically sets the title of the navigation bar for the current page.	1.5.0
setNavigationBarColor	Sets the color of the page navigation bar.	1.5.0
hideHomeButton	Hides the back to home button.	1.5.0
showNavigationBarLoading	Shows navigation bar loading animation on the current page.	1.5.1
hideNavigationBarLoading	Hides navigation bar loading animation on the current page.	1.5.1

Background

Name	Feature description	Minimum version
setBackgroundColor	Dynamically sets the background color of the window.	1.5.0
setBackgroundTextStyle	Dynamically sets the style for the drop-down background text or loading image.	1.5.0

Tab bar

Name	Feature description	Minimum version
showTabBar	Shows the tab bar.	1.5.0
hideTabBar	Hides the tab bar.	1.5.0
setTabBarStyle	Dynamically sets the overall style of the tab bar.	1.5.0
setTabBarItem	Dynamically sets the content of a certain item on the tab bar.	1.5.0
showTabBarRedDot	Shows the red dot in the top-right corner of a certain item on the tab bar.	1.5.0
hideTabBarRedDot	Hides the red dot in the top-right corner of a certain item on the tab bar.	1.5.0
setTabBarBadge	Adds text to the top-right corner of a certain item on the tab bar.	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

removeTabBarBadge	Removes the text from the top-right corner of a certain item on the tab bar.	1.5.0

Pull-to-refresh

Name	Feature description	Minimum version
startPullDownRefresh	Starts pull-to-refresh.	1.5.0
stopPullDownRefresh	Stops pull-to-refresh.	1.5.0

Scroll

Name	Feature description	Minimum version
pageScrollTo	Scrolls the page to the target position, supporting both selector and scroll distance methods for positioning.	1.5.0
ScrollViewContext.scrollIntoView	Scrolls an element to the top or bottom of the viewport.	1.5.0
ScrollViewContext.scrollTo	Scrolls an element to the specified position, precise to the pixel.	1.5.0

Animation

Name	Feature description	Minimum version
Name	Creates an animation instance.	1.5.0
createAnimation	Sets the background color.	1.5.0
Animation.backgroundColor	Sets the bottom value.	1.5.0
Animation.bottom	Exports the animation queue.	1.5.0
Animation.export	Sets the height.	1.5.0
Animation.height	Sets the left value.	1.5.0
Animation.left	Same as transform-function matrix.	1.5.0
Animation.matrix	Same as transform-function matrix3d.	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

Animation.matrix3d	Sets the opacity.	1.5.0
Animation.opacity	Sets the right value.	1.5.0
Animation.right	Rotates clockwise from the origin by a specified angle.	1.5.0
Animation.rotate	Rotates clockwise from a fixed axis by a specified angle.	1.5.0
Animation.rotate3d	Rotates clockwise from the x-axis by a specified angle.	1.5.0
Animation.rotateX	Rotates clockwise from the y-axis by a specified angle.	1.5.0
Animation.rotateY	Rotates clockwise from the z-axis by a specified angle.	1.5.0
Animation.rotateZ	Changes the scale.	1.5.0
Animation.scale	Changes the scale in a 3D scene.	1.5.0
Animation.scale3d	Change the x-axis scale.	1.5.0
Animation.scaleX	Change the y-axis scale.	1.5.0
Animation.scaleY	Change the z-axis scale.	1.5.0
Animation.scaleZ	Sets the x-axis and y-axis skew angle.	1.5.0
Animation.skew	Sets the x-axis skew angle.	1.5.0
Animation.skewX	Sets the y-axis skew angle.	1.5.0
Animation.skewY	Indicate a set of animations is complete.	1.5.0
Animation.step	Sets the top value.	1.5.0
Animation.top	Translation.	1.5.0
Animation.translate	Translates the X, Y, and Z coordinates.	1.5.0
Animation.translate3d	Translates the x-axis.	1.5.0
Animation.translateX	Translates the y-axis.	1.5.0
Animation.translateY	Translates the z-axis.	1.5.0
Animation.translateZ	Sets the width.	1.5.0
Animation.width		

©2013-2025 Tencent Cloud International Pte. Ltd.

Custom components

Name	Feature description	Minimum version
nextTick	Delays certain operations to the next time slice.	1.5.0

Menu

Name	Feature description	Minimum version
getMenuButtonBoundingClientRect	Gets the layout position information of the menu button (capsule button in the top right corner).	1.5.0

Window

Name	Feature description	Minimum version
onWindowResize	Listens for window size change event.	1.5.0
offWindowResize	Removes the listener for window size change event.	1.5.1

Font

Name	Feature description	Minimum version
loadFontFace	Dynamically loads web fonts.	1.5.0

Network

Make request

Name	Feature description	Minimum version
request	Makes an HTTPS network request.	1.5.0
RequestTask.abort	Aborts the request task.	1.5.0
RequestTask.onChunkReceived	Listens for the Transfer-Encoding Chunk Received event.	1.5.1

RequestTask.offChunkReceived	Removes the listener for the Transfer-Encoding Chunk Received event.	1.5.1
RequestTask.onHeadersReceived	Listens for the HTTP Response Header event.	1.5.0
RequestTask.offHeadersReceived	Removes the listener for the HTTP Response Header event.	1.5.0

Upload

Name	Feature description	Minimum version
uploadFile	Uploads local resources to the server.	1.5.0
UploadTask.abort	Aborts the upload task.	1.5.0
UploadTask.onProgressUpdate	Listens for upload progress change event.	1.5.0
UploadTask.offProgressUpdate	Removes the listener for the upload progress change event.	1.5.0
UploadTask.onHeadersReceived	Listens for the HTTP Response Header event.	1.5.0
UploadTask.offHeadersReceived	Removes the listener for the HTTP Response Header event.	1.5.0

Download

Name	Feature description	Minimum version
downloadFile	Downloads file resources to the local device. The client initiates an HTTPS GET request and returns the local temporary path of the file. The maximum allowed file size for a single download is 200 MB.	1.5.1
DownloadTask.abort	Aborts the download task.	1.5.1
DownloadTask.onProgressUpdate	Listens for download progress change event.	1.5.1
DownloadTask.offProgressUpdate	Removes the listener for download progress change event.	1.5.1
DownloadTask.onHeadersReceived	Listens for the HTTP Response Header event.	1.5.1
DownloadTask.offHeadersReceived	Removes the listener for the HTTP Response Header	1.5.1

©2013-2025 Tencent Cloud International Pte. Ltd.

	event.	

WebSocket

Name	Feature description	Minimum version
sendSocketMessage	Sends data using WebSocket connection. You need to call wx.connectSocket first and send data after the wx.onSocketOpen callback.	1.5.1
onSocketOpen	Listens for the WebSocket connection open event.	1.5.1
onSocketMessage	Listens for messages received from the server through WebSocket.	1.5.1
onSocketError	Listens for WebSocket error event.	1.5.1
onSocketClose	Listens for the WebSocket connection close event.	1.5.1
connectSocket	Creates a WebSocket connection.	1.5.1
closeSocket	Closes the last WebSocket connection created via wx.connectSocket (singleton mode).	1.5.1
SocketTask.close	Closes the WebSocket connection associated with the current SocketTask instance.	1.5.0
SocketTask.onClose	Listens for the WebSocket connection close event.	1.5.0
SocketTask.onError	Listens for WebSocket error event.	1.5.0
SocketTask.onMessage	Listens for messages received from the server through WebSocket.	1.5.0
SocketTask.onOpen	Listens for the WebSocket connection open event.	1.5.0
SocketTask.send	Sends data using WebSocket connection.	1.5.0

UDP communication

Name	Feature description	Minimum version
createUDPSocket	Creates a UDP Socket instance.	1.5.1
UDPSocket.bind	Binds to a system-assigned available port or a specified	1.5.1

	port number	
UDPSocket.close	Closes the UDP Socket instance.	1.5.1
UDPSocket.connect	Pre-connects to a specified IP and port, used together with the write method.	1.5.1
UDPSocket.onClose	Listen for the close event.	1.5.1
UDPSocket.offClose	Removes the listener for the close event.	1.5.1
UDPSocket.onError	Listens for the error event.	1.5.1
UDPSocket.offError	Removes the listener for the error event.	1.5.1
UDPSocket.onListening	Listens for the event when starting to listen for data packet messages.	1.5.1
UDPSocket.offListening	Removes the listener for the event when starting to listen for data packet messages.	1.5.1
UDPSocket.onMessage	Listens for the event of receiving a message.	1.5.1
UDPSocket.offMessage	Removes the listener for the event of receiving a message.	1.5.1
UDPSocket.send	Sends a message to the specified IP and port.	1.5.1
UDPSocket.setTTL	Sets the IP_TTL socket option to specify the maximum number of hops allowed for an IP packet during transmission.	1.5.16
UDPSocket.write	Same usage as UDPSocket.send.	1.5.16

TCP communication

Name	Feature description	Minimum version
createTCPSocket	Creates a TCP Socket instance.	1.5.1
TCPSocket.bindWifi	Binds the TCP Socket to the current Wi-Fi network, triggering the onBindWifi event upon success (Android only)	1.5.1
TCPSocket.close	Closes the connection.	1.5.1
TCPSocket.connect	Initiates a connection on the given socket.	1.5.1

TCPSocket.onClose	Listens for the event triggered when the socket is fully closed.	1.5.1
TCPSocket.offClose	Removes the listener for the event triggered when the socket is fully closed.	1.5.1
TCPSocket.onConnect	Listens for the event triggered when a socket connection is successfully established.	1.5.1
TCPSocket.offConnect	Removes the listener for the event triggered when a socket connection is successfully established.	1.5.1
TCPSocket.onError	Listens for the event triggered when an error occurs.	1.5.1
TCPSocket.offError	Removes the listener for the event triggered when an error occurs.	1.5.1
TCPSocket.onMessage	Listens for the event triggered when data is received.	1.5.1
TCPSocket.offMessage	Removes the listener for the event triggered when data is received.	1.5.1
TCPSocket.onBindWifi	Listens for the event triggered when a socket successfully binds to the current Wi-Fi network.	1.5.1
TCPSocket.offBindWifi	Removes the listener for the event triggered when a socket successfully binds to the current Wi-Fi network.	1.5.1
TCPSocket.write	Sends data on the socket.	1.5.1

mDNS

Name	Feature description	Minimum version
stopLocalServiceDiscovery	Stops searching for mDNS services.	1.5.1
startLocalServiceDiscovery	Starts searching for mDNS services on the local network.	1.5.1
offLocalServiceResolveFail	Removes the listener for the mDNS service resolution failure event.	1.5.1
onLocalServiceResolveFail	Listens for the mDNS service resolution failure event.	1.5.1
offLocalServiceLost	Removes the listener for the mDNS service lost event.	1.5.1
onLocalServiceLost	Listens for the mDNS service lost event.	1.5.1

©2013-2025 Tencent Cloud International Pte. Ltd.

offLocalServiceFound	Removes the listener for mDNS service found event.	1.5.1
onLocalServiceFound	Listens for the mDNS service found event.	1.5.1
offLocalServiceDiscoveryStop	Removes the listener for the mDNS service search stop event.	1.5.1
onLocalServiceDiscoveryStop	Listens for the mDNS service search stop event.	1.5.1

Payment

Name	Feature description	Minimum version
requestPayment	Mini program payment	2.2.2

Data cache

Name	Feature description	Minimum version
setStorage	Stores the data in local cache under the specified key.	1.5.0
setStorageSync	Synchronously stores the data in local cache under the specified key.	1.5.0
revokeBufferURL	Destroys data in memory based on the URL.	1.5.35
removeStorage	Removes the specified key from the local cache.	1.5.0
removeStorageSync	Synchronously removes the specified key from local cache.	1.5.0
getStorage	Asynchronously gets the content of the specified key from the local cache.	1.5.0
createBufferURL	Creates a unique URL in memory based on the provided buffer.	1.5.35
getStorageSync	Synchronously gets the content of the specified key from the local cache.	1.5.0
getStorageInfo	Asynchronously gets the information of the current	1.5.0

	storage.	
getStorageInfoSync	Synchronously gets the information of the current storage.	1.5.0
clearStorage	Clears the local data cache.	1.5.0
clearStorageSync	Synchronously clears local data cache.	1.5.0
batchSetStorage	Stores the data in batches to the specified key in the local cache.	1.5.16
batchSetStorageSync	Synchronously stores the data in batches to the specified key in the local cache.	1.5.16
batchGetStorage	Asynchronously gets the content of the specified key in batches from the local cache.	1.5.16
batchGetStorageSync	Synchronously gets the content of the specified key in batches from the local cache.	1.5.16

Data analysis

Name	Feature description	Minimum version
reportEvent	Reports the event.	1.5.16

Canvas

Canvas overview

Name	Feature description	Minimum version
createOffscreenCanvas	Creates an off-screen Canvas instance.	1.5.19
createCanvasContext	Creates a drawing context (CanvasContext) for the Canvas.	1.5.0
canvasToTempFilePath	Exports the content of the specified Canvas area to generate an image of the specified size.	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

canvasPutImageData	Draws pixel data onto the Canvas.	1.5.0
canvasGetImageData	Gets the pixel data of the specified area of the Canvas.	1.5.0
Image	Image object.	1.5.16
ImageData	ImageData object.	1.5.16
RenderingContext	Canvas rendering context.	1.5.16

canvasContext

Name	Feature description	Minimum version
CanvasContext.arc	Creates an arc.	1.5.0
CanvasContext.arcTo	Creates an arc using control points and radius.	1.5.0
CanvasContext.beginPath	Starts creating a new path.	1.5.0
CanvasContext.bezierCurveTo	Creates a cubic Bezier curve path.	1.5.0
CanvasContext.clearRect	Clears the contents within the specified rectangular area on the Canvas.	1.5.0
CanvasContext.clip	Clips a region of any shape and size from the original canvas.	1.5.0
CanvasContext.closePath	Closes a path.	1.5.0
CanvasContext.createCircularGradient	Creates a circular gradient.	1.5.0
CanvasContext.createLinearGradient	Creates a linear gradient.	1.5.0
CanvasContext.createPattern	Creates a pattern using the specified image, which can be repeated in the specified direction.	1.5.0
CanvasContext.draw	Renders the previously described paths, transformations, and styles in the drawing context onto the Canvas.	1.5.0
CanvasContext.drawImage	Draws an image onto the Canvas.	1.5.0
CanvasContext.fill	Fills the content in the current path.	1.5.0
CanvasContext.fillRect	Fills a rectangle.	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

CanvasContext.fillText	Draws the filled text on the Canvas.	1.5.0
CanvasContext.lineTo	Adds a new point and creates a line from the last specified point to the target point.	1.5.0
CanvasContext.measureText	Measures the dimensions of the specified text.	1.5.0
CanvasContext.moveTo	Moves the path to the specified point on the Canvas without creating lines.	1.5.0
CanvasContext.quadraticCurveTo	Creates a quadratic Bezier curve path.	1.5.0
CanvasContext.rect	Creates a rectangular path.	1.5.0
CanvasContext.restore	Restores the previously saved drawing context.	1.5.0
CanvasContext.rotate	Rotates the current coordinate axis clockwise around the origin.	1.5.0
CanvasContext.save	Saves the drawing context.	1.5.0
CanvasContext.scale	Scales the coordinates of paths created after the call.	1.5.0
CanvasContext.setFillStyle	Sets the fill color.	1.5.0
CanvasContext.setFontSize	Sets the font size.	1.5.0
CanvasContext.setGlobalAlpha	Sets the global alpha (transparency) of the brush.	1.5.0
CanvasContext.setLineCap	Sets the line endpoint style.	1.5.0
CanvasContext.setLineDash	Sets the style of the dashed line.	1.5.0
CanvasContext.setLineJoin	Sets the style of the line joins.	1.5.0
CanvasContext.setLineWidth	Sets the line width.	1.5.0
CanvasContext.setMiterLimit	Sets the maximum miter length.	1.5.0
CanvasContext.setShadow	Sets the shadow style.	1.5.0
CanvasContext.setStrokeStyle	Sets the stroke color.	1.5.0
CanvasContext.setTextAlign	Sets the text alignment.	1.5.0
CanvasContext.setTextBaseline	Sets the vertical alignment of text.	1.5.0
CanvasContext.setTransform	Resets (overwrites) the current transformation	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

	using a matrix.	
CanvasContext.stroke	Strokes the current path.	1.5.0
CanvasContext.strokeRect	Draws a non-filled rectangle.	1.5.0
CanvasContext.strokeText	Strokes a given text at the specified (x, y) position.	1.5.0
CanvasContext.transform	Multiplies the current transformation with the given matrix.	1.5.0
CanvasContext.translate	Translates the origin (0, 0) of the current coordinate system.	1.5.0

CanvasGradient

Name	Feature description	Minimum version
CanvasGradient.addColorStop	Adds a color stop to the gradient.	1.5.0

Canvas

Name	Feature description	Minimum version
Canvas.cancelAnimationFrame	Cancels an animation frame request previously scheduled with requestAnimationFrame.	1.5.16
Canvas.createImageData	Creates an ImageData object.	1.5.16
Canvas.createImage	Creates an image object.	1.5.16
Canvas.createPath2D	Creates a Path2D object.	1.5.16
Canvas.getContext	Returns the drawing context of the Canvas.	1.5.16
Canvas.requestAnimationFrame	Schedules a callback to be executed before the next repaint.	1.5.16
Canvas.toDataURL	Returns a data URL containing the image display.	1.5.16

OffscreenCanvas

Name	Feature description	Minimum version

OffscreenCanvas.createImage	Returns the drawing context of the OffscreenCanvas.	1.5.16
OffscreenCanvas.getContext	Creates an image object.	1.5.16

Path2D

Name	Feature description	Minimum version
Path2D.addPath	Adds a path to the current path.	1.5.16
Path2D.arc	Adds an arc to the path.	1.5.16
Path2D.arcTo	Adds an arc to the path with given control points.	1.5.16
Path2D.bezierCurveTo	Adds a cubic Bezier curve to the path.	1.5.16
Path2D.closePath	Closes the path by drawing a straight line back to the start.	1.5.16
Path2D.ellipse	Adds an elliptical arc to the path.	1.5.16
Path2D.lineTo	Adds a straight line to the path.	1.5.16
Path2D.moveTo	Moves the path starting point.	1.5.16
Path2D.quadraticCurveTo	Adds a quadratic Bezier curve to the path.	1.5.16
Path2D.rect	Adds a rectangle to the path.	1.5.16

Media

Map

Name	Feature description	Minimum version
createMapContext	Creates a MapContext object for the map.	1.5.0
MapContext.addArc	Adds an arc.	1.5.16
MapContext.addCustomLayer	Adds a custom layer.	1.5.16
MapContext.addGroundOverlay	Creates a custom image layer.	1.5.16
MapContext.addMarkers	Adds a marker.	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

MapContext.eraseLines	Erases or grays out lines that have been added to the map.	1.5.16
MapContext.fromScreenLocation	Gets the latitude and longitude corresponding to a point on the screen, with the origin at the top-left corner of the map.	1.5.0
MapContext.getCenterLocation	Gets the latitude and longitude of the current map center.	1.5.0
MapContext.getRegion	Gets the current visible region of the map.	1.5.0
MapContext.getRotate	Gets the current rotation angle of the map.	1.5.0
MapContext.getScale	Gets the current zoom level of the map.	1.5.0
MapContext.getSkew	Gets the current tilt angle of the map.	1.5.0
MapContext.includePoints	Zooms the view to display all latitudes and longitudes.	1.5.0
MapContext.initMarkerCluster	Initializes the configuration for marker clustering.	1.5.16
MapContext.moveAlong	Moves a marker along the specified path (used for track playback and other scenarios).	1.5.0
MapContext.moveToLocation	Moves the map center to the current location point. Requires the map component's show-location to be set to true.	1.5.0
MapContext.on	Listens for map events.	1.5.16
MapContext.openMapApp	Opens a map app for navigation selection.	1.5.0
MapContext.removeArc	Deletes an arc.	1.5.16
MapContext.removeCustomLayer	Removes a custom layer.	1.5.16
MapContext.removeGroundOverlay	Removes a custom image layer.	1.5.16
MapContext.removeMarkers	Removes a marker.	1.5.0
MapContext.setBoundary	Sets the display boundary of the map.	1.5.16
MapContext.setCenterOffset	Sets the offset for the map center point.	1.5.16
MapContext.setLocMarkerIcon	Sets the icon for the location marker.	1.5.16
MapContext.toScreenLocation	Gets the screen coordinates corresponding to the	1.5.16

©2013-2025 Tencent Cloud International Pte. Ltd.

	latitude and longitude.	
MapContext.translateMarker	Translates a marker with animation.	1.5.0
MapContext.updateGroundOverlay	Updates a custom image layer.	1.5.16

Image

Name	Feature description	Minimum version
chooseImage	Allows users to select images from the local photo album or take photos using the camera.	1.5.0
compressImage	Compresses an image with optional compression quality settings.	1.5.1
getImageInfo	Gets image information.	1.5.0
previewImage	Previews an image in full screen mode on a new page.	1.5.0
previewMedia	Allows users to preview images and videos.	1.5.1
saveImageToPhotosAlbum	Saves an image to the system album.	1.5.1

Real-time audio and video

Name	Feature	Minimum version
createLivePusherContext	Creates a LivePusherContext object for the live-pusher context.	1.5.0
createLivePlayerContext	Creates a LivePlayerContext object for the live-player context.	1.5.0
LivePlayerContext.exitfullscreen	Exits the full-screen mode.	1.5.0
LivePlayerContext.mute	Mutes the audio.	1.5.0
LivePlayerContext.pause	Pauses the playback.	1.5.0
LivePlayerContext.play	Starts the playback.	1.5.0
LivePlayerContext.requestFullScreen	Enters the full-screen mode.	1.5.0
LivePlayerContext.resume	Resumes the playback.	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

LivePlayerContext.snapshot	Captures a screenshot.	1.5.0
LivePlayerContext.stop	Stops the playback.	1.5.0
LivePusherContext.start	Starts pushing the stream and enables camera preview.	1.5.0
LivePusherContext.stop	Stops pushing the stream and disables camera preview.	1.5.0
LivePusherContext.pause	Pauses pushing the stream.	1.5.0
LivePusherContext.resume	Resumes pushing the stream.	1.5.0
LivePusherContext.switchCamera	Switches between front and rear cameras.	1.5.0
LivePusherContext.snapshot	Captures a snapshot.	1.5.0
LivePusherContext.toggleTorch	Toggles the flashlight.	1.5.0
LivePusherContext.playBGM	Plays background music.	1.5.0
LivePusherContext.stopBGM	Stops playing background music.	1.5.0
LivePusherContext.pauseBGM	Pauses playing background music.	1.5.0
LivePusherContext.resumeBGM	Resumes playing background music.	1.5.0
LivePusherContext.setBGMPosition	Sets the playback position of the background music.	1.5.0
LivePusherContext.setMICVolume	Sets the microphone volume.	1.5.0
LivePusherContext.setBGMVolume	Sets the background music volume.	1.5.0
LivePusherContext.setAudioReverbType	Sets the audio reverb type.	1.5.0
LivePusherContext.startPreview	Enables camera preview.	1.5.0
LivePusherContext.stopPreview	Disables camera preview.	1.5.0
LivePusherContext.startAudioRecord	Starts audio recording.	1.5.0
LivePusherContext.stopAudioRecord	Stops recording.	1.5.0

Audio

Name	Feature description	Minimum version

stopVoice	Stops the voice playback.	1.5.1
playVoice	Starts the voice playback.	1.5.1
pauseVoice	Pauses the voice playback.	1.5.1
setInnerAudioOption	Sets the playback options for InnerAudioContext.	1.5.0
getAvailableAudioSources	Gets the currently supported audio input sources.	1.5.1
createInnerAudioContext	Creates the InnerAudioContext object for the internal audio component.	1.5.0
AudioContext.pause	Pauses the audio playback.	1.5.1
AudioContext.play	Starts the audio playback.	1.5.1
AudioContext.seek	Seeks to a specified position in the audio.	1.5.1
AudioContext.setSrc	Sets the audio source URL.	1.5.1
InnerAudioContext.destroy	Destroys the current instance.	1.5.0
InnerAudioContext.offCanplay	Removes the event listener for when the audio can be played.	1.5.0
InnerAudioContext.offEnded	Removes the event listener for when the audio naturally ends.	1.5.0
InnerAudioContext.offError	Removes the event listener for audio playback errors.	1.5.0
InnerAudioContext.offPause	Removes the event listener for audio pause events.	1.5.0
InnerAudioContext.offPlay	Removes the event listener for audio play events.	1.5.0
InnerAudioContext.offSeeked	Removes the event listener for when the audio seek operation is completed.	1.5.0
InnerAudioContext.offSeeking	Removes the event listener for when the audio seek operation is in progress.	1.5.0
InnerAudioContext.offStop	Removes the event listener for audio stop events.	1.5.0
InnerAudioContext.offTimeUpdate	Removes the event listener for audio playback progress updates.	1.5.0
InnerAudioContext.offWaiting	Removes the event listener for audio loading events.	1.5.0
InnerAudioContext.onCanplay	Listens for the event when the audio can start playing.	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

InnerAudioContext.onEnded	Listens for the event when the audio naturally ends.	1.5.0
InnerAudioContext.onError	Listens for audio playback error event.	1.5.0
InnerAudioContext.onPause	Listens for audio pause event.	1.5.0
InnerAudioContext.onPlay	Listens for audio playback event.	1.5.0
InnerAudioContext.onSeeked	Listens for the event when audio seeking is completed.	1.5.0
InnerAudioContext.onSeeking	Listens for audio seeking event.	1.5.0
InnerAudioContext.onStop	Listens for audio stop event.	1.5.0
InnerAudioContext.onTimeUpdate	Listens for audio playback progress update event.	1.5.0
InnerAudioContext.onWaiting	Listens for audio loading event.	1.5.0
InnerAudioContext.pause	Pause the audio.	1.5.0
InnerAudioContext.play	Starts the playback.	1.5.0
InnerAudioContext.seek	Seeks to a specified position in the audio.	1.5.0
InnerAudioContext.stop	Stops the playback.	1.5.0

Video

Name	Description	Minimum version
saveVideoToPhotosAlbum	Saves a video to the system album.	1.5.0
createVideoContext	Creates a VideoContext object for the video component.	1.5.0
chooseVideo	Records a video or selects a video from the phone's album.	1.5.0
compressVideo	Compresses a video.	1.5.1
chooseMedia	Takes or selects images or videos from the phone's album.	1.5.1
VideoContext.exitFullScreen	Exits the full-screen mode.	1.5.0
VideoContext.exitBackgroundPlayback	Exits the background audio playback mode.	1.5.16

©2013-2025 Tencent Cloud International Pte. Ltd.

VideoContext.exitPictureInPicture	Exits the picture-in-picture mode. This method can be called on any page.	1.5.16
VideoContext.hideStatusBar	Hides the status bar, effective only in iOS full screen mode.	1.5.0
VideoContext.pause	Pauses the video.	1.5.0
VideoContext.play	Plays the video.	1.5.0
VideoContext.requestFullScreen	Enters the full-screen mode.	1.5.0
VideoContext.requestBackgroundPlayback	Enters the background audio playback mode.	1.5.16
VideoContext.seek	Seeks to a specified position in the audio.	1.5.0
VideoContext.sendDanmu	Sends an on-screen comment.	1.5.0
VideoContext.showStatusBar	Shows the status bar, effective only in iOS full screen mode.	1.5.0
VideoContext.stop	Stops the video.	1.5.0

Transparent video

Name	Description	Minimum version
createAnimationVideo	Creates an AnimationVideoContext object for the animation-video context.	1.5.0
AnimationVideoContext.play	Plays the video.	1.5.0
AnimationVideoContext.pause	Pauses the video.	1.5.0
AnimationVideoContext.seek	Seeks to a specified position in the audio.	1.5.0

Camera

Name	Feature description	Minimum version
createCameraContext	Creates a CameraContext object for the camera component.	1.5.0
CameraContext.onCameraFrame	Gets real-time frame data from the camera.	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

CameraContext.setZoom	Sets the zoom level.	1.5.0
CameraContext.startRecord	Starts video recording.	1.5.0
CameraContext.stopRecord	Stops video recording.	1.5.0
CameraContext.takePhoto	Takes a photo.	1.5.0
CameraFrameListener.start	Starts listening for frame data.	1.5.0
CameraFrameListener.stop	Stops listening for frame data.	1.5.0

Recording

Name	Feature	Minimum version
startRecord	Starts audio recording.	1.5.1
stopRecord	Stops audio recording.	1.5.1
getRecorderManager	Gets the globally unique RecorderManager.	1.5.0
RecorderManager.onError	Listens for the recording error event.	1.5.0
RecorderManager.onFrameRecorded	Listens for the event when a specified frame size of the file has been recorded.	1.5.0
RecorderManager.onInterruptionBegin	Listens for the event when recording is interrupted due to system resource occupation.	1.5.0
RecorderManager.onInterruptionEnd	Listens for the event when the recording interruption ends.	1.5.0
RecorderManager.onPause	Listens for the event when the recording pauses.	1.5.0
RecorderManager.onResume	Listens for the event when the recording resumes.	1.5.0
RecorderManager.onStart	Listens for the event when the recording starts.	1.5.0
RecorderManager.onStop	Listens for the event when the recording stops.	1.5.0
RecorderManager.pause	Pauses the recording.	1.5.0
RecorderManager.resume	Resumes the recording.	1.5.0
RecorderManager.start	Starts audio recording.	1.5.0
RecorderManager.stop	Pauses the recording.	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

Background audio

Name	Description	Minimum version
stopBackgroundAudio	Stops playing the music.	1.5.1
seekBackgroundAudio	Controls the playback progress of the background music.	1.5.1
playBackgroundAudio	Plays background music using the background player.	1.5.1
pauseBackgroundAudio	Pauses the music playback and listens for the event when music playback stops.	1.5.1
onBackgroundAudioStop	Listens for the background music stop event.	1.5.1
onBackgroundAudioPlay	Listens for background music playback event.	1.5.1
onBackgroundAudioPause	Listens for background music pause event.	1.5.1
getBackgroundAudioPlayerState	Gets the background music playback status.	1.5.1
getBackgroundAudioManager	Gets the globally unique background audio manager.	1.5.1
BackgroundAudioManager.onCanplay	Listens for the event when the background audio can start playing.	1.5.16
BackgroundAudioManager.onEnded	Listens for the event when the background audio naturally ends.	1.5.16
BackgroundAudioManager.onError	Listens for background audio playback error event.	1.5.16
BackgroundAudioManager.onNext	Listens for events when the user taps the next track on the system music playback panel (iOS only).	1.5.16
BackgroundAudioManager.onPause	Listens for the background audio pause event.	1.5.16
BackgroundAudioManager.onPlay	Listens for the background audio playback event.	1.5.16
BackgroundAudioManager.onPrev	Listens for events when the user taps the previous track on the system music playback	1.5.16

©2013-2025 Tencent Cloud International Pte. Ltd.

	panel (iOS only).	
BackgroundAudioManager.onSeeked	Listens for the event when the background audio seek operation is completed.	1.5.16
BackgroundAudioManager.orSeeking	Listens for the event when the background audio seek operation starts.	1.5.16
BackgroundAudioManager.onStop	Listens for the background audio stop event.	1.5.16
BackgroundAudioManager.onTimeUpdate	Listens for the background audio playback progress update event.	1.5.16
BackgroundAudioManager.onWaiting	Listens for audio loading event.	1.5.16
BackgroundAudioManager.pause	Pauses the background music.	1.5.16
BackgroundAudioManager.play	Plays the background music.	1.5.16
BackgroundAudioManager.seek	Seeks to a specified position in the background audio.	1.5.16
BackgroundAudioManager.stop	Stops the background music.	1.5.16

Rich text

Name	Description	Minimum version
EditorContext	An EditorContext instance.	1.5.16
EditorContext.blur	Blurs the editor and hides the keyboard.	1.5.16
EditorContext.clear	Clears the content of the editor.	1.5.16
EditorContext.format	Modifies the format.	1.5.16
EditorContext.getContents	Gets the content of the editor.	1.5.16
EditorContext.getSelectionText	Gets the plain text content of the selected area in the editor.	1.5.16
EditorContext.insertDivider	Inserts a divider.	1.5.16
EditorContext.insertImage	Inserts an image.	1.5.16
EditorContext.insertText	Sets a segment of text, replacing the current selection.	1.5.16

©2013-2025 Tencent Cloud International Pte. Ltd.

EditorContext.redo	Redo	1.5.16
EditorContext.removeFormat	Clears the format of the current selection.	1.5.16
EditorContext.scrollIntoView	Scrolls the editor cursor into the visible area of the window.	1.5.16
EditorContext.setContents	Initializes the content of the editor.	1.5.16
EditorContext.undo	Undo	1.5.1

File

File overview

Name	Description	Minimum version
saveFile	Saves a file to the local storage.	1.5.0
removeSavedFile	Deletes local cache files.	1.5.0
openDocument	Opens a document in a new page.	1.5.0
getSavedFileList	Gets the list of locally cached files saved by the mini program.	1.5.0
getSavedFileInfo	Gets the information of a local file.	1.5.0
getFileSystemManager	Gets the globally unique file manager.	1.5.0
getFileInfo	Gets the file information.	1.5.0
ReadResult	File read result.	1.5.16
WriteResult	File write result.	1.5.16

FileSystemManager

API	Description	Minimum version
FileSystemManager.access	Determines whether a file or directory exists.	1.5.0
FileSystemManager.accessSync	Synchronously checks if a file/directory	1.5.0

	exists.	
FileSystemManager.appendFile	Appends content to the end of a file.	1.5.0
FileSystemManager.appendFileSync	Synchronously appends content to the end of a file.	1.5.0
FileSystemManager.close	Closes a file.	1.5.16
FileSystemManager.closeSync	Synchronously closes a file.	1.5.16
FileSystemManager.copyFile	Copies a file.	1.5.0
FileSystemManager.copyFileSync	Synchronously copies a file.	1.5.0
FileSystemManager.fstat	Gets status information of a file.	1.5.16
FileSystemManager.fstatSync	Synchronously gets status information of a file.	1.5.16
FileSystemManager.ftruncate	Truncates file contents.	1.5.16
FileSystemManager.ftruncateSync	Synchronously truncates file contents.	1.5.16
FileSystemManager.getFileInfo	Gets the information of local temporary file or local cache file in this mini program.	1.5.0
FileSystemManager.getSavedFileList	Gets the list of locally cached files saved by the mini program.	1.5.0
FileSystemManager.mkdir	Creates a directory.	1.5.0
FileSystemManager.mkdirSync	Synchronously creates a directory.	1.5.0
FileSystemManager.open	Opens a file and returns a file descriptor.	1.5.16
FileSystemManager.openSync	Synchronously opens a file and returns a file descriptor.	1.5.16
FileSystemManager.read	Reads a file.	1.5.16
FileSystemManager.readSync	Synchronously reads a file.	1.5.16
FileSystemManager.readCompressedFile	Reads the content of a local file with a specified compression type.	1.5.16
FileSystemManager.readCompressedFileSync	Synchronously reads the content of a local file with a specified compression type.	1.5.16

FileSystemManager.readdir	Reads the list of files in the directory.	1.5.0
FileSystemManager.readdirSync	Synchronously reads the list of files in the directory.	1.5.0
FileSystemManager.readFile	Reads local file contents.	1.5.0
FileSystemManager.readFileSync	Synchronously reads local file contents.	1.5.0
FileSystemManager.readZipEntry	Reads files in a compressed package.	1.5.16
FileSystemManager.removeSavedFile	Deletes saved local cache files in this mini program.	1.5.0
FileSystemManager.rename	Renames a file.	1.5.0
FileSystemManager.renameSync	Synchronously renames a file.	1.5.0
FileSystemManager.rmdir	Deletes a directory.	1.5.0
FileSystemManager.rmdirSync	Synchronously deletes a directory.	1.5.0
FileSystemManager.saveFile	Saves a temporary file to a local storage.	1.5.0
FileSystemManager.saveFileSync	Synchronously saves a temporary file to a local storage.	1.5.0
FileSystemManager.stat	Gets a file Stats object.	1.5.0
FileSystemManager.statSync	Synchronously gets a file Stats object.	1.5.0
FileSystemManager.truncate	Truncates file contents.	1.5.16
FileSystemManager.truncateSync	Synchronously truncates file contents.	1.5.16
FileSystemManager.unlink	Deletes a file.	1.5.0
FileSystemManager.unlinkSync	Synchronously deletes a file.	1.5.0
FileSystemManager.unzip	Decompresses a file.	1.5.0
FileSystemManager.write	Writes to a file.	1.5.16
FileSystemManager.writeSync	Synchronously writes to a file.	1.5.16
FileSystemManager.writeFile	Writes a file.	1.5.0
FileSystemManager.writeFileSync	Synchronously writes a file.	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

Stats

API	Description	Minimum version
Stats.isDirectory	Checks whether the current file is a directory.	1.5.0
Stats.isFile	Checks whether the current file is a normal file.	1.5.0

Open APIs

Login

Name	Feature description	Minimum version
login	Calls the API to get the login credential (code).	1.5.1
checkSession	Checks if the login session has expired.	1.5.1
requestSubscribeMessage	Opens the client mini program subscription message interface	2.1.1

Account information

Name	Feature description	Minimum version
getAccountInfoSync	Gets the current account information.	1.5.1

User information

Name	Feature description	Minimum version
getUserProfile	Gets user information.	1.5.1
getUserInfo	Gets user information, requires user authorization for scope.userInfo.	1.5.1
userInfo	User information.	1.5.0

Settings

Name	Feature description	Minimum version
AuthSetting	User’s authorization settings information.	1.5.0
getSetting	Gets the user's current settings.	1.5.0
openSetting	Opens the mini program settings interface and returns the user's operation results.	1.5.0

Biometric authentication

Name	Feature description	Minimum version
checkIsSoterEnrolledInDevice	Checks whether biometric information is entered into the device.	1.5.0
checkIsSupportSoterAuthentication	Verifies if the device supports biometric authentication.	1.5.0
startSoterAuthentication	Starts the biometric authentication.	1.5.0

Authorization

Name	Feature description	Minimum version
authorize	Initiates an authorization request to the user in advance.	1.5.0

Location

Name	Feature description	Minimum version
getLocation	Gets the current geographic location and speed.	1.5.0
choosePoi	Opens the POI list to select the location. Supports both fuzzy (city-level) and precise location selection.	1.5.0
chooseLocation	Opens the map and selects a location.	1.5.0
stopLocationUpdate	Stops listening for real-time geographic location	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

	changes, halting message reception in both foreground and background.	
startLocationUpdateBackground	Enables receiving location messages when the mini program enters both foreground and background, requires user authorization.	1.5.0
startLocationUpdate	Enables receiving location messages when the mini program enters the frontend.	1.5.0
openLocation	Uses the built-in map to view the location.	1.5.0
onLocationChangeError	Listens for the event triggered when the continuous location API returns a failure.	1.5.0
onLocationChange	Listens for real-time geographic location change events, used in conjunction with startLocationUpdateBackground and startLocationUpdate.	1.5.0
offLocationChangeError	Removes the listener for failures returned by the continuous location API.	1.5.0
offLocationChange	Removes the listener for real-time geographic location change events.	1.5.0
getFuzzyLocation	Gets the current fuzzy geographic location.	1.5.0

Device

General-purpose Bluetooth API

Name	Feature description	Minimum version
stopBluetoothDevicesDiscovery	Stops searching for nearby Bluetooth peripheral devices.	1.5.0
startBluetoothDevicesDiscovery	Starts searching for nearby Bluetooth peripheral devices.	1.5.0
openBluetoothAdapter	Initializes the Bluetooth module.	1.5.1
getConnectedBluetoothDevices	Gets connected Bluetooth devices based on the main service UUID.	1.5.1

getBluetoothDevices	Gets all Bluetooth devices discovered during the active period of the Bluetooth module.	1.5.1
getBluetoothAdapterState	Gets the state of the Bluetooth adapter.	1.5.1
closeBluetoothAdapter	Closes the Bluetooth module.	1.5.1
onBluetoothDeviceFound	Listens for the event when a new device is discovered.	1.5.0
offBluetoothDeviceFound	Removes all listeners for the event when a new device is discovered.	1.5.0
onBluetoothAdapterStateChange	Listens for the event when the Bluetooth adapter state changes.	1.5.0
offBluetoothAdapterStateChange	Removes all listeners for the event when the Bluetooth adapter state changes.	1.5.0
makeBluetoothPair	The Bluetooth pairing API, which is supported only on Android.	1.5.0
isBluetoothDevicePaired	Checks if a Bluetooth device is paired, supported only on Android.	1.5.0

Bluetooth Low Energy (BLE) central device

Name	Feature description	Minimum version
writeBLECharacteristicValue	Writes binary data to the characteristic value of a Bluetooth Low Energy (BLE) device.	1.5.1
readBLECharacteristicValue	Reads binary data from the characteristic value of a Bluetooth Low Energy (BLE) device.	1.5.1
setBLEMTU	Negotiates and sets the Maximum Transmission Unit (MTU) for Bluetooth Low Energy.	1.5.1
getBLEMTU	Gets the Maximum Transmission Unit (MTU) for Bluetooth Low Energy.	1.5.1
onBLEMTUChange	Listens for the event when the Maximum Transmission Unit (MTU) for Bluetooth Low Energy changes (triggered only on Android).	1.5.1
offBLEMTUChange	Removes the listener for the Maximum	1.5.1

	Transmission Unit (MTU) change event.	
onBLEConnectionStateChange	Listens for the event when the connection status of a Bluetooth Low Energy device changes.	1.5.1
offBLEConnectionStateChange	Removes the listener for changes in the connection state of Bluetooth Low Energy devices.	1.5.1
onBLECharacteristicValueChange	Listens for changes in the characteristic values of Bluetooth Low Energy devices.	1.5.1
offBLECharacteristicValueChange	Remove all listeners for changes in the characteristic values of Bluetooth Low Energy devices.	1.5.1
notifyBLECharacteristicValueChange	Enables the notify feature for characteristic value changes in Bluetooth Low Energy devices, subscribing to the characteristic.	1.5.1
getBLEDeviceServices	Gets all services of a Bluetooth Low Energy device.	1.5.1
getBLEDeviceRSSI	Gets the signal strength of a Bluetooth Low Energy device.	1.5.1
getBLEDeviceCharacteristics	Gets all characteristics of a specific service in a Bluetooth Low Energy device.	1.5.1
createBLEConnection	Connects to a Bluetooth Low Energy device.	1.5.1
closeBLEConnection	Disconnects from a Bluetooth Low Energy device.	1.5.1

Bluetooth Low Energy (BLE) peripheral device

Name	Feature description	Minimum version
onBLEPeripheralConnectionStateChanged	Listens for events when the current peripheral device is connected or disconnected.	1.5.1
offBLEPeripheralConnectionStateChanged	Removes the listener for events when the current peripheral device is connected or disconnected.	1.5.1
createBLEPeripheralServer	Establishes a local server as a Bluetooth Low Energy (BLE)	1.5.1

©2013-2025 Tencent Cloud International Pte. Ltd.

	peripheral server.	
BLEPeripheraServer.addService	Adds a service.	1.5.1
BLEPeripheraServer.removeService	Removes a service.	1.5.1
BLEPeripheraServer.startAdvertising	Starts advertising the locally created peripheral device.	1.5.1
BLEPeripheraServer.stopAdvertising	Stops advertising the locally created peripheral device.	1.5.1
BLEPeripheraServer.writeCharacteristicValue	Writes a binary data value to the specified characteristic and notifies the connected host.	1.5.1
BLEPeripheraServer.onCharacteristicWriteRequest	Listens for the event when a connected device requests to write the characteristic value of the current peripheral device.	1.5.1
BLEPeripheraServer.offCharacteristicWriteRequest	Removes the listener for the event when a connected device requests to write the characteristic value of the current peripheral device.	1.5.1
BLEPeripheraServer.onCharacteristicReadRequest	Listens for the event when a connected device requests to read the characteristic value of the current peripheral device.	1.5.1
BLEPeripheraServer.offCharacteristicReadRequest	Removes the listener for the event when a connected device requests to read the characteristic value of the current peripheral device.	1.5.1
BLEPeripheraServer.onCharacteristicSubscribed	Listens for the characteristic subscription event (supported only on iOS).	1.5.1
BLEPeripheraServer.offCharacteristicSubscribed	Removes the listener for the characteristic subscription event (supported only on iOS).	1.5.1
BLEPeripheraServer.onCharacteristicUnsubscribed	Listens for the characteristic unsubscription event (supported only on iOS).	1.5.1

©2013-2025 Tencent Cloud International Pte. Ltd.

BLEPeripheraServer.offCharacteristicUnsubscribed	Removes the listener for the characteristic unsubscription event (supported only on iOS).	1.5.1

Bluetooth - Beacon

Name	Feature description	Minimum version
stopBeaconDiscovery	Stops searching for nearby Beacon devices.	1.5.1
startBeaconDiscovery	Starts searching for nearby Beacon devices.	1.5.1
onBeaconUpdate	Listens for the Beacon device update event (only one listener can be registered).	1.5.1
offBeaconUpdate	Remove all listeners for the Beacon device update event.	1.5.1
onBeaconServiceChange	Listens for the Beacon service status change event (only one listener can be registered).	1.5.1
offBeaconServiceChange	Removes all listeners for the Beacon service status change event.	1.5.1
getBeacons	Gets all discovered Beacon devices.	1.5.1
BeaconInfo	Beacon device information.	1.5.1

NFC

getNFCAdapter

Name	Feature description	Minimum version
getNFCAdapter	Gets the NFC instance.	1.5.0

NFCAdapter

Name	Feature description	Minimum version
NFCAdapter.startDiscovery	Starts the device.	1.5.0
NFCAdapter.stopDiscovery	Stops the device.	1.5.0

NFCAdapter.getNdef	Gets the Ndef instance, which supports reading and writing NDEF data on NFC tags in NDEF format.	1.5.0
NFCAdapter.getNfcA	Gets the NfcA instance, which supports reading and writing according to the NFC-A (ISO 14443-3A) standard.	1.5.0
NFCAdapter.getNfcB	Gets the NfcB instance, which supports reading and writing according to the NFC-B (ISO 14443-3B) standard.	1.5.0
NFCAdapter.getNfcF	Gets the NfcF instance, which supports reading and writing according to the NFC-F (JIS 6319-4) standard.	1.5.0
NFCAdapter.getNfcV	Gets the NfcV instance, which supports reading and writing according to NFC-V (ISO 15693) standard.	1.5.0
NFCAdapter.getIsoDep	Gets the IsoDep instance, which supports reading and writing according to ISO-DEP (ISO 14443-4) standard.	1.5.0
NFCAdapter.getMifareClassic	Gets the MifareClassic instance, which supports reading and writing MIFARE Classic tags.	1.5.0
NFCAdapter.getMifareUltralight	Gets the MifareUltralight instance, which supports reading and writing MIFARE Ultralight tags.	1.5.0
NFCAdapter.onDiscovered	Listens for the NFC tag discovery event.	1.5.0
NFCAdapter.offDiscovered	Removes the listener for the NFC tag discovery event.	1.5.0

IsoDep

Name	Feature description	Minimum version
IsoDep.connect	Connects to the NFC tag.	1.5.0
IsoDep.close	Disconnects from the NFC tag.	1.5.0
IsoDep.setTimeout	Sets the timeout duration.	1.5.0
IsoDep.isConnected	Checks if the NFC tag is connected.	1.5.0
IsoDep.getMaxTransceiveLength	Gets the maximum transceive length.	1.5.0

IsoDep.transceive	Sends data.	1.5.0
IsoDep.getHistoricalBytes	Gets the historical bytes.	1.5.0

MifareClassic

Name	Feature description	Minimum version
MifareClassic.connect	Connects to the NFC tag.	1.5.0
MifareClassic.close	Disconnects from the NFC tag.	1.5.0
MifareClassic.setTimeout	Sets the timeout duration.	1.5.0
MifareClassic.isConnected	Checks if the NFC tag is connected.	1.5.0
MifareClassic.getMaxTransceiveLength	Gets the maximum transceive length.	1.5.0
MifareClassic.transceive	Sends data.	1.5.0

MifareUltralight

Name	Feature description	Minimum version
MifareUltralight.connect	Connects to the NFC tag.	1.5.0
MifareUltralight.close	Disconnects from the NFC tag.	1.5.0
MifareUltralight.setTimeout	Sets the timeout duration.	1.5.0
MifareUltralight.isConnected	Checks if the NFC tag is connected.	1.5.0
MifareUltralight.getMaxTransceiveLength	Gets the maximum transceive length.	1.5.0
MifareUltralight.transceive	Sends data.	1.5.0

Ndef

Name	Feature description	Minimum version
Ndef.connect	Connects to the NFC tag.	1.5.0
Ndef.close	Disconnects from the NFC tag.	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

Ndef.setTimeout	Sets the timeout duration.	1.5.0
Ndef.isConnected	Checks if the NFC tag is connected.	1.5.0
Ndef.offNdefMessage	Removes the listener for Ndef messages.	1.5.0
Ndef.onNdefMessage	Listens for Ndef messages.	1.5.0
Ndef.writeNdefMessage	Rewrites the content of an Ndef tag.	1.5.0

NfcA

Name	Feature description	Minimum version
NfcA.connect	Connects to the NFC tag.	1.5.0
NfcA.close	Disconnects from the NFC tag.	1.5.0
NfcA.setTimeout	Sets the timeout duration.	1.5.0
NfcA.isConnected	Checks if the NFC tag is connected.	1.5.0
NfcA.getMaxTransceiveLength	Gets the maximum transceive length.	1.5.0
NfcA.transceive	Sends data.	1.5.0
NfcA.getAtqa	Gets ATQA information.	1.5.0
NfcA.getSak	Gets SAK information.	1.5.0

NfcB

Name	Feature description	Minimum version
NfcB.connect	Connects to the NFC tag.	1.5.0
NfcB.close	Disconnects from the NFC tag.	1.5.0
NfcB.setTimeout	Sets the timeout duration.	1.5.0
NfcB.isConnected	Checks if the NFC tag is connected.	1.5.0
NfcB.getMaxTransceiveLength	Gets the maximum transceive length.	1.5.0
NfcB.transceive	Sends data.	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

NfcF

Name	Feature description	Minimum version
NfcF.connect	Connects to the NFC tag.	1.5.0
NfcF.close	Disconnects from the NFC tag.	1.5.0
NfcF.setTimeout	Sets the timeout duration.	1.5.0
NfcF.isConnected	Checks if the NFC tag is connected.	1.5.0
NfcF.getMaxTransceiveLength	Gets the maximum transceive length.	1.5.0
NfcF.transceive	Sends data.	1.5.0

NfcV

Name	Feature description	Minimum version
NfcV.connect	Connects to the NFC tag.	1.5.0
NfcV.close	Disconnects from the NFC tag.	1.5.0
NfcV.setTimeout	Sets the timeout duration.	1.5.0
NfcV.isConnected	Checks if the NFC tag is connected.	1.5.0
NfcV.getMaxTransceiveLength	Gets the maximum transceive length.	1.5.0
NfcV.transceive	Sends data.	1.5.0

Wi-Fi

Name	Feature description	Minimum version
onWifiConnected	Listens for the event when Wi-Fi is connected.	1.5.1
offWifiConnected	Removes the listener for the event when Wi-Fi is connected.	1.5.1
stopWifi	Turns off the Wi-Fi module.	1.5.0
startWifi	Initializes the Wi-Fi module.	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

onGetWifiList	Listens for the event when Wi-Fi list data is obtained.	1.5.0
offGetWifiList	Removes the listener for the event when Wi-Fi list data is obtained.	1.5.0
getConnectedWifi	Gets information about the connected Wi-Fi.	1.5.0
WifiInfo	Wi-Fi information.	1.5.1

Calendar

Name	Feature description	Minimum version
addPhoneRepeatCalendar	Adds a recurring event to the system calendar.	1.5.1
addPhoneCalendar	Adds an event to the system calendar.	1.5.1

Clipboard

Name	Feature description	Minimum version
setClipboardData	Sets the content of the system clipboard.	1.5.1
getClipboardData	Gets the content of the system clipboard.	1.5.1

Network

Name	Feature description	Minimum version
onNetworkStatusChange	Listens for network status change event.	1.5.1
getNetworkType	Gets the network type.	1.5.1
offNetworkStatusChange	Removes the listener for network status change event.	1.5.1

Encryption

Name	Feature description	Minimum version
getRandomValues	Gets cryptographically secure pseudo-random	1.5.1

©2013-2025 Tencent Cloud International Pte. Ltd.

	values.	

Keyboard

Name	Feature description	Minimum version
onKeyboardHeightChange	Listens for keyboard height change event.	1.5.1
hideKeyboard	Manually hides the keyboard after it is raised by focusing on input or textarea elements.	1.5.1
offKeyboardHeightChange	Removes the listener for keyboard height change event.	1.5.1
getSelectedTextRange	Gets the cursor position within the input field after focusing on elements like input or textarea.	1.5.1

Phone

Name		Feature description	Minimum version
makePhoneCall		Makes a phone call.	1.5.1

Accelerometer

Name	Feature description	Minimum version
startAccelerometer	Starts listening for accelerometer data.	1.5.1
stopAccelerometer	Stops listening for acceleration data.	1.5.1
onAccelerometerChange	Listens for accelerometer data change event.	1.5.1
offAccelerometerChange	Removes the listener for accelerometer data changes event.	1.5.1

Compass

Name	Feature description	Minimum version
startCompass	Starts listening for compass data.	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

stopCompass	Stops listening for compass data event.	1.5.0
onCompassChange	Listens for compass data change event.	1.5.0
offCompassChange	Removes the listener for compass data change event.	1.5.0

Gyroscope

Name	Feature description	Minimum version
startGyroscope	Starts listening for gyroscope data.	1.5.1
stopGyroscope	Stops listening for gyroscope data.	1.5.1
onGyroscopeChange	Listens for gyroscope data change event.	1.5.1
offGyroscopeChange	Removes the listener for gyroscope data change event.	1.5.1

Scan code

Name	Feature description	Minimum version
scanCode	Opens the client’s QR code scanning interface to scan the QR code.	1.5.0

SMS

Name	Feature description	Minimum version
sendSms	Opens the phone's SMS sending interface.	1.5.1

Vibration

Name	Feature description	Minimum version
vibrateShort	Causes the phone to vibrate for a short duration (15 ms).	1.5.1
vibrateLong	Causes the phone to vibrate for a long duration (400	1.5.1

©2013-2025 Tencent Cloud International Pte. Ltd.

	ms).	

Contacts

Name	Feature description	Minimum version
chooseContact	Opens the phone's contact list to select a contact.	1.5.1
addPhoneContact	Adds a contact to the phone's contact list.	1.5.1

Battery

Name	Feature description	Minimum version
getBatteryInfoSync	Synchronously gets battery information of the device.	1.5.1
getBatteryInfo	Gets the device's battery information.	1.5.1

Screen

Name	Feature description	Minimum version
setVisualEffectOnCapture	Sets the screen's visual effect during screenshot/recording.	1.5.1
setScreenBrightness	Sets the screen brightness.	1.5.1
setKeepScreenOn	Sets whether to keep the screen on.	1.5.1
onUserCaptureScreen	Listens for user-initiated screenshot event.	1.5.1
onScreenRecordingStateChanged	Listens for user-initiated screen recording event.	1.5.1
offUserCaptureScreen	User-initiated screenshot event.	1.5.1
offScreenRecordingStateChanged	Removes the listener for user-initiated screen recording event.	1.5.1
getScreenRecordingState	Checks if the user is recording the screen.	1.5.1
getScreenBrightness	Gets the screen brightness.	1.5.1

Device motion

Name	Feature description	Minimum version
stopDeviceMotionListening	Stops listening for device motion change event.	1.5.1
startDeviceMotionListening	Starts listening for device motion change event.	1.5.1
onDeviceMotionChange	Listens for device motion change event.	1.5.1
offDeviceMotionChange	Removes the listener for device motion change event.	1.5.1

Memory

Name	Feature description	Minimum version
onMemoryWarning	Listens for the low memory alert event.	1.5.1
offMemoryWarning	Removes the listener for the low memory alert event.	1.5.1

Accessibility

Name	Feature description	Minimum version
checkIsOpenAccessibility	Checks whether the visual accessibility feature is enabled.	1.5.1

WXML

WXML overview

Name	Feature description	Minimum version
createSelectorQuery	Returns a SelectorQuery object instance.	1.5.0
createIntersectionObserver	Creates and returns an IntersectionObserver object instance.	1.5.0

IntersectionObserve

Name	Feature description	Minimum version
IntersectionObserve.relativeTo	Uses a selector to specify a node as one of the reference areas.	1.5.0
IntersectionObserve.relativeToViewport	Specifies the viewport as one of the reference areas.	1.5.0
IntersectionObserve.disconnect	Stops listening, and the callback function will no longer be triggered.	1.5.0
IntersectionObserve.observe	Specifies the target node and starts listening for intersection status changes.	1.5.0

NodesRef

Name	Feature description	Minimum version
NodesRef.fields	Gets relevant information about the node.	1.5.0
NodesRef.boundingClientRect	Adds a query request for the node's layout position.	1.5.0
NodesRef.scrollOffset	Adds a query request for the node's scroll position.	1.5.0
NodesRef.context	Adds a query request for the node's Context object.	1.5.0
NodesRef.node	Gets the Node instance.	1.5.0

SelectorQuery

Name	Feature description	Minimum version
SelectorQuery.exec	Executes all requests.	1.5.0
SelectorQuery.in	Changes the selection scope of the selector to within a custom component.	1.5.0
SelectorQuery.select	Selects the first node that matches the selector within the current page.	1.5.0
SelectorQuery.selectAll	Selects all nodes that match the selector within the current page.	1.5.0
SelectorQuery.selectViewport	Selects the viewport.	1.5.0

©2013-2025 Tencent Cloud International Pte. Ltd.

meidaQueryObserver

Name	Feature description	Minimum version
MediaQueryObserver.disconnect	Stops listening.	1.5.16
MediaQueryObserver.observe	Starts listening for media query changes.	1.5.16

Custom APIs

Name	Feature description	Minimum version
invokeNativePlugin	Custom APIs.	1.5.0

Ad

Name	Feature description	Minimum version
createRewardedVideoAd	Rewarded ad.	2.2.3

©2013-2025 Tencent Cloud International Pte. Ltd.

Basic

Last updated：2025-05-26 10:47:42

canIUse

This API is called using Boolean wx.canIUse(string schema).

Feature description: Checks whether the APIs, callbacks, parameters, and components of the mini program are supported in the current version.
Parameter and description: String schema
Use the format `${API}.${method}.${param}.${options}` or `${component}.${attribute}.${option}` to call.
Parameter description:

Parameter	Description
${API}	API name.
${method}	Call method. Valid values: return, success, object, callback.
${param}	Parameter or return value.
${options}	Optional values of the parameter.
${component}	Component name.
${attribute}	Component property.
${option}	Optional value of the component property.

Return value: Boolean, indicating whether the feature is available in the current version.
Example:

```
wx.canIUse('openBluetoothAdapter')
wx.canIUse('getSystemInfoSync.return.screenWidth')
wx.canIUse('getSystemInfo.success.screenWidth')
wx.canIUse('showToast.object.image')
wx.canIUse('onCompassChange.callback.direction')
```

©2013-2025 Tencent Cloud International Pte. Ltd.

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

wx.canIUse('live-player')
wx.canIUse('text.selectable')
wx.canIUse('button.open-type.contact')
```

env

This API is called using wx.env.

Feature description: Environment variables.

Parameter and description:string USER_DATA_PATH, the user directory path in the file system (local path).

base64ToArrayBuffer

This API is called using ArrayBuffer wx.base64ToArrayBuffer(string base64).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Converts a Base64 string into an ArrayBuffer object.

Parameter and description:string base64, the Base64 string to be converted into an ArrayBuffer object.

Return value: ArrayBuffer. An ArrayBuffer object.

Example:

```
const base64 = 'CxYh'
const arrayBuffer = wx.base64ToArrayBuffer(base64)
```

arrayBufferToBase64

This API is called using string wx.arrayBufferToBase64(ArrayBuffer arrayBuffer).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Converts an ArrayBuffer object into a Base64 string.

Parameter and description: ArrayBuffer arrayBuffer, the ArrayBuffer object to be converted into a Base64 string.

Return value:string, a Base64 string.

Example:

```
const arrayBuffer = new Uint8Array([11, 22, 33])
const base64 = wx.arrayBufferToBase64(arrayBuffer)
```

System

getSystemInfo

This API is called using wx.getUserInfo(Object object).

Feature description: Gets system information.

Parameter and description: Object Object.

Property	Type	Default value	Required	Description
success	Function	-	False	Callback function for successful API calls.
fail	Function	-	False	Callback function for failed API calls.
complete	Function	-	False	Callback function executed after API call ends (regardless of success or failure).

Res callback result of Object.success

Property	Type	Description
brand	String	Device brand.
model	String	Device model.
pixelRatio	Number	Device pixel ratio.
screenWidth	Number	Screen width in px.
screenHeight	Number	Screen height in px.

windowWidth	Number	Available window width in px.
windowHeight	Number	Available window height in px.
statusBarHeight	Number	Status bar height in px.
language	String	Language.
version	String	Version number.
system	String	Operating system and version.
platform	String	Client platform. Valid values: iOS: including iPhone and iPad Android devtools: TCSAS developer tools
fontSizeSetting	number	User font size in px.
SDKVersion	String	Base library version of the client.
AppPlatform	String	App platform.
safeArea	Object	Safe area in portrait mode.
theme	String	Current system theme. `Value: Light or dark.` Only available if `"darkmode":true` Otherwise, it is undefined.

res.safeArea structure:

Property	Type	Description
left	Number	X-coordinate of the top-left corner of the safe area.
right	Number	X-coordinate of the bottom-right corner of the safe area.
top	Number	Y-coordinate of the top-left corner of the safe area.
bottom	Number	Y-coordinate of the bottom-right corner of the safe area.
width	Number	Width of the safe area in logical pixels.
height	Number	Height of the safe area in logical pixels.

Example:

```
wx.getSystemInfo({
  success(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)
  }
})
```

```
try {
  const 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
}
```

getSystemInfoSync

This API is called using Object wx.getSystemInfoSync().

Feature description: Synchronously performs the same feature as wx.getSystemInfo.

Res callback result of Object.success

Property	Type	Description
brand	String	Device brand.
model	String	Device model.
pixelRatio	Number	Device pixel ratio.
screenWidth	Number	Screen width in px.
screenHeight	Number	Screen height in px.
windowWidth	Number	Available window width in px.

windowHeight	Number	Available window height in px.
statusBarHeight	Number	Status bar height in px.
language	String	Language.
version	String	Version number.
system	String	Operating system and version.
platform	String	Client platform. Valid values: iOS: including iPhone and iPad Android devtools: TCSAS developer tools
fontSizeSetting	number	User font size in px.
AppPlatform	String	App platform.
safeArea	Object	Safe area in portrait mode.
theme	String	Current system theme. `Value: Light or dark.` Only available if `"darkmode":true` Otherwise, it is undefined. (Not supported in mini games)

res.safeArea structure:

Property	Type	Description
left	Number	X-coordinate of the top-left corner of the safe area.
right	Number	X-coordinate of the bottom-right corner of the safe area.
top	Number	Y-coordinate of the top-left corner of the safe area.
bottom	Number	Y-coordinate of the bottom-right corner of the safe area.
width	Number	Width of the safe area in logical pixels.
height	Number	Height of the safe area in logical pixels.

Example:

```
wx.getSystemInfo({
  success(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)
  }
})
```

```
try {
  const 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
}
```

getSystemInfoAsync

This API is called using wx.getSystemInfoAsync(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Asynchronously gets system information. This feature is enabled only on certain versions of host client. On unsupported clients, system information is got synchronously.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.success callback function parameter: Object res

Property	Type	Description

brand	string	Device brand.
model	string	Device model. New models may show as Unknown for a while until adapted.
pixelRatio	number	Device pixel ratio.
screenWidth	number	Screen width in px.
screenHeight	number	Screen height in px.
windowWidth	number	Available window width in px.
windowHeight	number	Available window height in px.
statusBarHeight	number	Task status bar height in px.
language	string	Language set in the app.
version	string	App version number.
system	string	Operating system and version.
platform	string	Client platform. Valid values: iOS: including iPhone and iPad Android devtools: IDE
fontSizeSetting	number	User font size in px.
SDKVersion	string	Base library version of the client.
safeArea	object	Safe area in portrait mode. Some devices do not have a safe area concept and will not return the safeArea field. Developers need to handle this compatibility. For return values, see the table **Return values of safeArea**
theme	string	The current system theme. The value can only be obtained when darkmode: true is set in global configuration. Otherwise, it is undefined (not supported in mini games). Valid values: Light: bright mode Dark: dark mode

Return values of safeArea

Structural property	Type	Description

left	number	X-coordinate of the top-left corner of the safe area.
right	number	X-coordinate of the bottom-right corner of the safe area.
top	number	Y-coordinate of the top-left corner of the safe area.
bottom	number	Y-coordinate of the bottom-right corner of the safe area.
width	number	Width of the safe area in logical pixels.
height	number	Height of the safe area in logical pixels.

Example:

```
wx.getSystemInfoAsync({
  success (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)
  }
})
```

getWindowInfo

This API is called using Object wx.getWindowInfo().

Feature description: Gets window information.

Parameter and description: Object.

Property	Type	Description
pixelRatio	number	Device pixel ratio.
screenWidth	number	Screen width in px.

screenHeight	number	Screen height in px.
windowWidth	number	Available window width in px.
windowHeight	number	Available window height in px.
statusBarHeight	number	Status bar height in px.
safeArea	object	Safe area in portrait mode. Some devices do not have a safe area concept and will not return the safeArea field. Developers need to handle this compatibility.
screenTop	number	The y-coordinate of the top edge of the window.

Return values of safeArea

Structural property	Type	Description
left	number	X-coordinate of the top-left corner of the safe area.
right	number	X-coordinate of the bottom-right corner of the safe area.
top	number	Y-coordinate of the top-left corner of the safe area.
bottom	number	Y-coordinate of the bottom-right corner of the safe area.
width	number	Width of the safe area in logical pixels.
height	number	Height of the safe area in logical pixels.

Example:

```
const windowInfo = wx.getWindowInfo()

console.log(windowInfo.pixelRatio)
console.log(windowInfo.screenWidth)
console.log(windowInfo.screenHeight)
console.log(windowInfo.windowWidth)
console.log(windowInfo.windowHeight)
console.log(windowInfo.statusBarHeight)
console.log(windowInfo.safeArea)
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
console.log(windowInfo.screenTop)
```

getSystemSetting

This API is called using Object wx.getSystemSetting().

Note:

This API is supported in mini programs but not in mini games.

Feature description: Gets device settings.

Parameter and description: Object.

Property	Type	Description
bluetoothEnabled	boolean	Bluetooth permission.
locationEnabled	boolean	Location permission.
wifiEnabled	boolean	Wi-Fi permission.
deviceOrientation	string	Device orientation. Valid values: Portrait: Portrait mode Landscape: Landscape mode

Example:

```
const systemSetting = wx.getSystemSetting()

console.log(systemSetting.bluetoothEnabled)
console.log(systemSetting.deviceOrientation)
console.log(systemSetting.locationEnabled)
console.log(systemSetting.wifiEnabled)
```

openSystemBluetoothSetting

This API is called using wx.openSystemBluetoothSetting(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Redirects to the system's Bluetooth settings page. Only supported on Android.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
wx.openSystemBluetoothSetting({
  success (res) {
    console.log(res)
  }
})
```

openAppAuthorizeSetting

This API is called using wx.openAppAuthorizeSetting(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Redirects to the system host client's authorization management page.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
wx.openAppAuthorizeSetting({
  success (res) {
```

```
    console.log(res)
  }
})
```

getRendererUserAgent

This API is called using Promise wx.getRendererUserAgent(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Gets the UserAgent of the web-view container where the mini program is running.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.success callback function parameter:string userAgent, UserAgent.

Return value: Promise.<string>.

Example:

```
wx.getRendererUserAgent().then(userAgent => console.log(userAgent))
wx.getRendererUserAgent({
    success(res) {
      console.log(res.userAgent)
  }
})
```

getDeviceInfo

This API is called using Object wx.getDeviceInfo().

Feature description: Gets the basic information of the device.

Return value: Object.

Property	Type	Description
abi	string	Host app's binary interface type (Android only).
deviceAbi	string	Device's binary interface type (Android only).
benchmarkLevel	number	Device performance level (Android only). Valid values: -2 or 0: Mini games are not supported on the device. -1: Performance unknown. >=1: Device performance value (the max value is currently less than 50). The higher the value, the better the device performance.
brand	string	Device brand.
model	string	Device model. New models may show as Unknown for a while until adapted.
system	string	Operating system and version.
platform	string	Client platform.
cpuType	string	Device CPU model (supported on Android only).**The GPU model can be obtained using WebGLRenderingContext.getExtension('WEBGL_debug_renderer_info').**
menorySize	string	Device memory size in MB.

Example:

```
const deviceInfo = wx.getDeviceInfo()

console.log(deviceInfo.abi)
console.log(deviceInfo.benchmarkLevel)
console.log(deviceInfo.brand)
console.log(deviceInfo.model)
console.log(deviceInfo.platform)
console.log(deviceInfo.system)
```

getAppBaseInfo

This API is called using Object wx.getAppBaseInfo().

Feature description: Gets basic information of the host client.

Parameter and description: Object.

Property	Type	Description
SDKVersion	string	Base library version of the host client.
enableDebug	boolean	Whether debugging has been enabled. You can open the menu in the upper right corner, tap Enable debugging or using wx.setEnableDebug to enable debugging.
host	Object	The host environment where the current mini program is running.
language	string	Language set on the host client.
version	string	Host client version number.
theme	string	The current system theme. Valid values: light: Bright mode, or dark: Dark mode. The value can only be obtained when "darkmode":true is set in global configuration. Otherwise, it is undefined (not supported in mini games).

Example:

```
const appBaseInfo = wx.getAppBaseInfo()

console.log(appBaseInfo.SDKVersion)
console.log(appBaseInfo.enableDebug)
console.log(appBaseInfo.host)
console.log(appBaseInfo.language)
console.log(appBaseInfo.version)
console.log(appBaseInfo.theme)
```

getAppAuthorizeSetting

This API is called using Object wx.getAppAuthorizeSetting().

Feature description: Gets permission info of the host client.

Parameter and description: Object.

Property	Type	Description
albumAuthorized	'authorized'/'denied'/'not determined'	Gets the album permission info of the host client. (iOS only).
bluetoothAuthorized	'authorized'/'denied'/'not determined'	Gets the Bluetooth permission info of the host client. (iOS only).

cameraAuthorized	'authorized'/'denied'/'not determined'	Gets the camera permission info of the host client.
locationAuthorized	'authorized'/'denied'/'not determined'	Gets the location permission info of the host client.
locationReducedAccuracy	boolean	Location accuracy. true indicates approximate location, while false indicates precise location (iOS only).
microphoneAuthorized	'authorized'/'denied'/'not determined'	Gets the microphone permission info of the host client.
notificationAuthorized	'authorized'/'denied'/'not determined'	Gets the notification permission info of the host client.
notificationAlertAuthorized	'authorized'/'denied'/'not determined'	Gets the reminder permission info of the host client. (iOS only).
notificationBadgeAuthorized	'authorized'/'denied'/'not determined'	Gets the notification badge permission info of the host client. (iOS only).
notificationSoundAuthorized	'authorized'/'denied'/'not determined'	Gets the sound notification permission info of the host client. (iOS only).
phoneCalendarAuthorized	'authorized'/'denied'/'not determined'	Gets the calendar permission info of the host client.

Example:

```
const appAuthorizeSetting = wx.getAppAuthorizeSetting()

console.log(appAuthorizeSetting.albumAuthorized)
console.log(appAuthorizeSetting.bluetoothAuthorized)
console.log(appAuthorizeSetting.cameraAuthorized)
console.log(appAuthorizeSetting.locationAuthorized)
console.log(appAuthorizeSetting.locationReducedAccuracy)
console.log(appAuthorizeSetting.microphoneAuthorized)
console.log(appAuthorizeSetting.notificationAlertAuthorized)
console.log(appAuthorizeSetting.notificationAuthorized)
console.log(appAuthorizeSetting.notificationBadgeAuthorized)
console.log(appAuthorizeSetting.notificationSoundAuthorized)
console.log(appAuthorizeSetting.phoneCalendarAuthorized)
```

©2013-2025 Tencent Cloud International Pte. Ltd.

Update

getUpdateManager

This API is called using UpdateManager wx.getUpdateManager().

Feature description: Gets **the globally unique** version update manager, used for managing the mini program updates.
Return value: UpdateManager object.

UpdateManager

.applyUpdate

This method is called using UpdateManager.applyUpdate().

Feature description: Forces the mini program to restart and use the new version. It is called after the new version of the mini program is downloaded (i.e. upon receipt of the `onUpdateReady` callback).

.onCheckForUpdate

This method is called using UpdateManager.onCheckForUpdate(function listener).

Feature description: Requests the backend to check for updates. This process is automatically triggered during the mini program's cold start, so developers don't need to initiate it manually.
Parameter and description:function callback. A callback function for the event that requests the backend to check for updates.

Property	Type	Description
hasUpdate	Boolean	Checks if a new version is available.

.onUpdateFailed

This method is called using UpdateManager.onUpdateFailed(function listener).

©2013-2025 Tencent Cloud International Pte. Ltd.

Feature description: Listens for mini program update failure event. When a new version of the mini program is available, the client automatically initiates the download (no developer action required). If the download fails (possibly due to network issues), this callback is triggered.
Parameter and description:function callback. A callback function for the event when the mini program update fails.

.onUpdateReady

This method is called using UpdateManager.onUpdateReady(function listener).

Feature description: Listens for mini program update event. The client initiates the download (no developer triggering required). Once the download is successful, this callback is triggered.
Parameter and description:function callback. A callback function for the event when a new version of the mini program is available.
Example:

```
const updateManager = wx.getUpdateManager()

updateManager.onCheckForUpdate(function (res) {
  // Callback after requesting new version information
  console.log(res.hasUpdate)
})

updateManager.onUpdateReady(function () {
  wx.showModal({
    title: 'Update prompt',
    content: 'A new version is ready. Would you like to restart the application?
    success(res) {
      if (res.confirm) {
        // The new version has been downloaded. Call applyUpdate to apply the new v
        updateManager.applyUpdate()
      }
    }
  })
})

updateManager.onUpdateFailed(function () {
  // New version download failed
})
```

Mini program

Lifecycle

getLaunchOptionsSync

This API is called using Object wx.getLaunchOptionsSync().

Note:

In some versions, `referrerInfo` may return `undefined.` It is recommended to use `options.referrerInfo && options.referrerInfo.appId` .

Feature description: Gets the parameters when the mini program launches.

Return value: Object, launch parameters.

Property	Type	Description
path	String	The path used to launch the mini program. **(for mini programs only).**
scene	Number	The scene value when the mini program is launched, default is 1001.
query	Object	The query parameters used to launch the mini program.
referrerInfo	Object	The referrer of the information, returned when entering a mini program from another mini program or an app. Otherwise, it returns {}.
forwardMaterials	Array.<Object>	An array of opened file information.

referrerInfo structure

Property	Type	Description
appId	String	The referrer's AppID for the mini program or app.
extraData	Object	Data passed from the referrer of the mini program.

Note:

extraData is the extended parameter passed from the client when the mini program launches.

For iOS: It is the paramsStr parameter passed in the startUMiniAppWithAppID method.

```
    [[TMFMiniAppSDKManager sharedInstance] startUpMiniAppWithAppID:@"mpbz0uqwzddj5z

    }];
```

For Android: It is the params field of the MiniStartLinkOptions object.

```
MiniStartLinkOptions miniStartOptions = new MiniStartLinkOptions(); miniStartOption
public static void startMiniApp(Activity activity, String appId, MiniStartOptions o
}
```

forwardMaterials structure

Property	Type	Description
type	string	The MIME type of the file.
name	Object	File name.
path	string	File path.
size	number	File size.

Scenes with valid `referrerInfo` returned

Property	Type	Description
1037	Opens the mini program from another mini program	The referrer of the mini program.
1038	Returns from another mini program	The referrer of the mini program.

getEnterOptionsSync

This API is called using Object wx.getEnterOptionsSync().

Note:

In some versions, `referrerInfo` may return `undefined` It is recommended to use `options.referrerInfo && options.referrerInfo.appId` .

Feature description: Gets the parameters when the mini program launches. If it is a cold start, the return value is consistent with the callback parameter of App.onLaunch. If it is a hot start, the return value is consistent with App.onShow.

Return value: Object

Property	Type	Description
path	string	The path used to launch the mini program (code package path). **(for mini programs only).**
scene	number	The scene value used to launch the mini program.
query	Object	The query parameters used to launch the mini program.
shareTicket	string	shareTicket
referrerInfo	Object	The information of the referrer. Returned when the mini program is accessed from another mini program or an app. Otherwise, {} is returned. For details, see the note above the table.
forwardMaterials	Array.<Object>	An array of opened file information.

referrerInfo structure

Property	Type	Description
appId	String	The referrer's AppID for the mini program or app.
extraData	Object	Data passed from the referrer of the mini program.

forwardMaterials structure

Property	Type	Description
type	String	The MIME type of the file.
name	Object	File name.
path	String	File path, or URL if using web-view component.
size	Number	File size.

Scenes with valid referrerInfo returned

Property	Type	Description

1037	Opens the mini program from another mini program	The referrer of the mini program.
1038	Returns from another mini program	The referrer of the mini program.

App-level events

onError

This API is called using wx.onError(function callback).

Note:
This API is supported in mini programs but not in mini games.
Feature description: Listens for mini program error event, such as script errors or API call failures.
Parameter and description:function callback. A callback function for mini program error event.
Response parameters: string error, error information, including the stack.

offError

This API is called using wx.offError(function callback).

Note:
This API is supported in mini programs but not in mini games.
Feature description: Cancels listening for mini program errors.
Parameter and description:function callback. A callback function for mini program error event.

onThemeChange

This API is called using wx.onThemeChange(function listener).

Note:
This event is triggered only when darkmode:true is set in the global configuration.
This API is supported in mini programs but not in mini games.
Feature description: Listens for system theme change event.

Parameter and description: function listener, the listener for system theme change event.

Response parameters: Object res

Property	Valid values and description	Type	Description
theme	Dark: Dark theme Light: Light theme	String	The current theme of the system, with the value is light or dark.

offThemeChange

This API is called using wx.offThemeChange(function listener).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Removes the listener for system theme change event.

Parameter and description:function listener. A listener passed to onThemeChange. If not provided, all listeners will be removed.

Example:

```
const listener = function (res) { console.log(res) }

wx.onThemeChange(listener)
wx.offThemeChange(listener) // Must pass the same function object used in onThemeCh
```

onPageNotFound

This API is called using wx.onPageNotFound(function listener).

Note:

This API is supported in mini programs but not in mini games.

Note:

Developers can perform page redirection within the callback, but it must be handled synchronously. Asynchronous processing (such as asynchronous execution with setTimeout) is invalid.

If the developer does not call wx.onPageNotFound to bind the listener, and does not state App.onPageNotFound, the host client will display a native "page not found" error page when attempting to navigate to a non-existent page.

If the callback redirects to another page that does not exist, the host client will display the native "page not found" error page and will not trigger the callback a second time.

©2013-2025 Tencent Cloud International Pte. Ltd.

Feature description: Listens for the event when the mini program attempts to open a non-existent page. This event is triggered at the same time as the App.onPageNotFound.
Parameter and description: Object res parameter, function listener, the listener for the event when the mini program attempts to open a non-existent page.

Property	Type	Description
path	string	The path of the non-existent page (code package path).
query	Object	Query parameter for opening a non-existent page.
isEntryPage	boolean	Whether it is the first page of this launch (for example, when entering from a shared link, the first page is the share page configured by the developer).

offPageNotFound

This API is called using wx.offPageNotFound(function listener).

Note:
This API is supported in mini programs but not in mini games.
Feature description: Removes the listener for the event when the mini program attempts to open a non-existent page.
Parameter and description: function listener. A listener passed to onPageNotFound. If not provided, all listeners will be removed.
Example:

```
const listener = function (res) { console.log(res) }

wx.onPageNotFound(listener)
wx.offPageNotFound(listener) // Must pass the same function object used in onPageNo
```

onAppShow

This API is called using wx.onAppShow(function listener).

Note:
This API is supported in mini programs but not in mini games.
Note:

In some versions, referrerInfo may return undefined if not present. It is recommended to check using `options.referrerInfo && options.referrerInfo.appId` .

Feature description: Listens for the event when the mini program switches to the frontend. This event is triggered at the same time as the App.onShow.

Parameter and description: Object res parameter, function listener, the listener for the event when the mini program switches to the frontend.

Property	Type	Description
path	string	The path used to launch the mini program (code package path).
scene	number	The scene value used to launch the mini program.
query	Object	The query parameters used to launch the mini program.
shareTicket	string	shareTicket
referrerInfo	Object	The information of the referrer. Returned when the mini program is accessed from another mini program or an app. Otherwise, {} is returned. For details, see the note above the table.
forwardMaterials	Array.<Object>	An array of opened file information.

referrerInfo structure

Property	Type	Description
appId	string	The referrer's AppID for the mini program or app.
extraData	Object	Data passed from the referrer of the mini program.

forwardMaterials structure

Property	Type	Description
type	String	The MIME type of the file.
name	Object	File name.
path	String	File path, or URL if using web-view component.
size	Number	File size.

Scenes with valid referrerInfo returned

Scene value	Scene	Description
1037	Opens the mini program from another mini program	The referrer of the mini program.
1038	Returns from another mini program	The referrer of the mini program.

offAppShow

This API is called using wx.offAppShow(function listener).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Removes the listener for the event when the mini program switches to the backend.

Parameter and description: function listener. A listener passed to onAppShow. If not provided, all listeners will be removed.

Example:

```
const listener = function (res) { console.log(res) }

wx.onAppShow(listener)
wx.offAppShow(listener) // Must pass the same function object used in onAppShow
```

onAppHide

This API is called using wx.onAppHide(function listener).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Listens for the event when the mini program switches to the backend. This event is triggered at the same time as the App.onHide.

Parameter and description: function listener. A listener for events of mini program switching to the backend.

offAppHide

The API is called using wx.offAppHide(function listener).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Removes the listener for the event when the mini program switches to the backend.

Parameter and description: function listener. A listener passed to onAppHide. If not provided, all listeners will be removed.

Example:

```
const listener = function (res) { console.log(res) }

wx.onAppHide(listener)
wx.offAppHide(listener) // Must pass the same function object used in onAppHide
```

onShow

This API is called using wx.onShow(function listener).

Note:

This API is only supported in mini games.

Feature description:A listener for the event when the mini game switches to the frontend.

Parameter and description: Object res parameter, function listener, a listener for the event when the mini game switches to the frontend.

Property	Type	Description
scene	number	Scene value.
query	Object	Query parameter.
shareTicket	string	shareTicket
referrerInfo	Object	The information of the referrer. Returned when the mini program is accessed from another mini program or an app.

referrerInfo structure

Property	Type	Description
appId	string	The referrer's AppID for the mini program or app.
extraData	Object	Data passed from the referrer of the mini program.

offShow

This API is called using wx.offShow(function listener).

Note:

This API is only supported in mini games.

Feature description: Removes the listener for the event when the mini game switches to the frontend.

Parameter and description:function listener. A listener passed to onShow. If not provided, all listeners will be removed.

Example:

```
const listener = function (res) { console.log(res) }

wx.onShow(listener)
wx.offShow(listener) // Must pass the same function object used in onShow
```

onHide

This API is called using wx.onHide(function listener).

Note:

This API is only supported in mini games.

Feature description: Listens for the event when the mini game switches to the backend. This event is triggered when the user locks the screen or press the Home button to return to the home screen.

Parameter and description:function listener. A listener for the event when the mini game switches to the backend.

offHide

The API is called using wx.offHide(function listener).

Note:

This API is only supported in mini games.

Feature description: Removes the listener for the event when the mini game switches to the backend.

Parameter and description:function listener. A listener passed to onHide. If not provided, all listeners will be removed.

Example:

```
const listener = function (res) { console.log(res) }
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
wx.onHide(listener)
wx.offHide(listener) // Must pass the same function object used in onHide
```

onUnhandledRejection

This API is called using wx.onUnhandledRejection(function listener).

Note:

This API is supported in mini programs but not in mini games.

Note:

All unhandledRejection events can be captured by this listener, but only those of the error type will trigger an alert in the mini program backend.

Feature description: Listens for unhandled Promise rejection event.

Parameter and description: Object res parameter, function listener, the listener for unhandled Promise rejection event.

Property	Type	Description
reason	string	Rejection reason, typically an Error object.
promise	Promise.<any>	The rejected Promise object.

offUnhandledRejection

This API is called using wx.offUnhandledRejection(function listener).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Removes the listener for unhandled Promise rejection event.

Parameter and description: function listener. A listener passed to onUnhandledRejection. If not provided, all listeners will be removed.

Example

```
const listener = function (res) { console.log(res) }

wx.onUnhandledRejection(listener)
wx.offUnhandledRejection(listener) // Must pass the same function object used in on
```

Debug

setEnableDebug

This API is called using wx.setEnableDebug(Object object).

Feature description: Sets whether to enable the debugging. It is also effective in the released version.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
enableDebug	boolean	-	True	Whether to enable debugging.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
// Enable debugging
wx.setEnableDebug({
  enableDebug: true
})

// Disable debugging
wx.setEnableDebug({
  enableDebug: false
})
```

Note:

You can also enable debugging in the development version or preview version first, and then switch to the released version where you can see the vConsole.

getRealtimeLogManager

This API is called using RealtimeLogManager wx.getRealtimeLogManager().

Feature description: Gets the real-time log manager object.

Example

```
// For the mini program
const logger = wx.getRealtimeLogManager()
logger.info({str: 'hello world'}, 'info log', 100, [1, 2, 3])
logger.error({str: 'hello world'}, 'error log', 100, [1, 2, 3])
logger.warn({str: 'hello world'}, 'warn log', 100, [1, 2, 3])


// For the plugin. Supported on the basic library 2.16.0 and later versions, and on
const logManager = wx.getRealtimeLogManager()
const logger = logManager.tag('plugin-log1')
logger.info('key1', 'value1')
logger.error('key2', {str: 'value2'})
logger.warn('key3', 'value3')
```

getLogManager

Th is API is called using LogManager wx.getLogManager(Object object).

Feature description: Gets the log manager object.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
level	number	0	False	The valid values are 0 and1, where 0 indicates the calls of lifecycle functions of `App` and `Page,` , as well as function calls under the namespace, will be logged. A value of 1 means they will not be logged. The default value is 0.

Return value: LogManager.

Example:

```
const logger = wx.getLogManager({level: 1})
logger.log({str: 'hello world'}, 'basic log', 100, [1, 2, 3])
logger.info({str: 'hello world'}, 'info log', 100, [1, 2, 3])
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
logger.debug({str: 'hello world'}, 'debug log', 100, [1, 2, 3])
logger.warn({str: 'hello world'}, 'warn log', 100, [1, 2, 3])
```

console

The console is a global object that can be accessed directly, with a set of features as follows. You can print logs in the IDE debugging panel and use vConsole to output logs on the client.

debug

> This method is called using console.debug().

Feature description: Prints debug logs in the debugging panel.

Parameter and description:any ... args, log content, with unlimited quantity.

error

> This method is called using console.error().

Feature description: Prints error logs in the debugging panel.

Parameter and description:any ... args, log content, with unlimited quantity.

group

> This method is called using console.group().

Note:

This feature is only effective within the tool, and is implemented as an empty function in the vConsole.

Feature description: Creates a new group in the debugging panel. Subsequent outputs will be indented, indicating that they belong to the current group. The group ends when console.groupEnd is called.

Parameter and description:string label, a label for the group (optional).

groupEnd

> This method is called using console.groupEnd().

Note:

This feature is only effective within the tool, and is implemented as an empty function in the vConsole.
Feature description: Ends a group created by console.group.
info

> This method is called using console.info.

Feature description: Prints info logs in the debugging panel.
Parameter and description:any ... args, log content, with unlimited quantity.
log

> This method is called using console.log().

Feature description: Prints log logs in the debugging panel.
Parameter and description:any ... args, log content, with unlimited quantity.
warn

> This method is called using console.warn().

Note:
Due to the limited features of vConsole and the support for console features varied among clients, it is recommended that developers only use the features provided in this document for mini programs.
Feature description: Prints warn logs in the debugging panel.
Parameter and description:any ... args, log content, with unlimited quantity.

LogManager

The log manager instance, which can be accessed using getLogManager, with the set of features as follows.
debug

> This method is called using LogManager.debug().

Feature description: Writes debug logs.
Parameter and description: Object|Array|number|string ... args, log content, with unlimited quantity. The total size of the parameters for each call can not exceed 100 KB.

info

This method is called using LogManager.info().

Feature description: Writes info logs.
Parameter and description: Object|Array.\\ |number|string ...args, log content, with unlimited quantity. The total size of the parameters for each call can not exceed 100 KB.
log

This method is called using LogManager.log().

Feature description: Writes log messages.
Parameter and description: Object|Array.\\ |number|string ...args, log content, with unlimited quantity. The total size of the parameters for each call can not exceed 100 KB.
warn

This method is called using LogManager.warn().

Note:
A maximum of 5 MB of log content can be saved. Once it exceeds 5MB, the old log content will be deleted.
Users can use the `<Button>` component's `open-type="feedback"` feature to upload the printed logs. Developers can view the feedback on the User feedback page on the mini program management backend.
The base library defaults to write the calls of lifecycle functions of `App` and `Page,` , and calls of functions under the namespace into the log.
Feature description: Writes warn logs.
Parameter and description: Object|Array.\\ |number|string ...args, log content, with unlimited quantity. The total size of the parameters for each call can not exceed 100 KB.
addFilterMsg

This method is called using RealtimeLogManager.addFilterMsg(string msg).

Feature description: Adds filter keywords.

Parameter and description:string msg, used as the parameter for the setFilterMsg. It is used to set multiple filter keywords.

error

This method is called using RealtimeLogManager.error().

Feature description: Writes error logs.

Parameter and description: Object|Array.<any> |number|string ...args, log content, with unlimited quantity. The total size of the parameters for each call can not exceed 5 KB.

getCurrentState

This method is called using Object RealtimeLogManager.getCurrentState().

Note:

The base library adds some structured data when reporting logs. If overflow occurs, warn logs are also provided. Therefore, the current usage information obtained by this method may be slightly larger than expected.

Feature description: Real-time logs aggregate and report the logs cached within a certain period. If the content cached exceeds the limit, it will be discarded. This feature can be used to obtain the cache remaining space.

Return value: Object

Property	Type	Description
size	number	The space already used in the current cache in bytes.
maxSize	number	The maximum available space in the current cache in bytes.
logCount	number	The number of log entries in the current cache.
maxLogCount	number	The maximum number of log entries that can be stored in the current cache.

in

This method is called using RealtimeLogManager.in(Page pageInstance).

Feature description: Sets the page parameter for real-time logs.

Parameter and description: Page pageInstance. A page instance.

info

This method is called using RealtimeLogManager.info().

Feature description: Writes info logs.
Parameter and description: Object|Array.<any> |number|string ...args, log content, with unlimited quantity. The total size of the parameters for each call can not exceed 5 KB.
setFilterMsg

This method is called using RealtimeLogManager.setFilterMsg(string msg).

Feature description: Sets the filter keywords.
Parameter and description:string msg, filter keyword, which can not exceed 1 KB. You can search the logs based on the settings in the mini program management backend.
warn

This method is called using RealtimeLogManager.warn().

Feature description: Writes warn logs.
Parameter and description: Object|Array.<any> |number|string ...args, log content, with unlimited quantity. The total size of the parameters for each call can not exceed 5 KB.

RealtimeTagLogManager

addFilterMsg

This method is called using RealtimeTagLogManager.addFilterMsg(string msg).

Feature description: Adds filter keywords.
Parameter and description:string msg, used as the parameter for the setFilterMsg. It is used to set multiple filter keywords.
error

©2013-2025 Tencent Cloud International Pte. Ltd.

This method is called using RealtimeTagLogManager.error(string key, Object | Array.<any> | number | string value).

Feature description: Writes error logs.
Parameter and description:
string key, the key of the log.
Object|Array.<any> |number|string value, the value of the log. The total size of the parameters for each call can not exceed 5 KB.
info

This method is called using RealtimeTagLogManager.info(string key, Object Array.<any> number string value).

Feature description: Writes info logs.
Parameter and description:
string key, the key of the log.
Object|Array.<any> |number|string value, the value of the log. The total size of the parameters for each call can not exceed 5 KB.
setFilterMsg

This method is called using RealtimeTagLogManager.setFilterMsg(string msg).

Feature description: Sets the filter keywords.
Parameter and description:string msg, filter keyword, which can not exceed 1 KB. You can search the logs based on the settings in the mini program management backend.
warn

This method is called using RealtimeTagLogManager.warn(string key, Object|Array|number|string value).

Feature description: Writes warn logs.
Parameter and description:
string key, the key of the log.

©2013-2025 Tencent Cloud International Pte. Ltd.

Object|Array|number|string value, the value of the log. The total size of the parameters for each call can not exceed 5 KB.

Performance

Note:

Only supported in mini games.

triggerGC

> This API is called using wx.triggerGC().

Feature description: Accelerates triggering JavaScriptCore garbage collection. The timing of garbage collection is controlled by JavaScriptCore, and it cannot guarantee that garbage collection is triggered immediately after the call.

getPerformance

> This API is called using wx.getPerformance().

Feature description: Gets performance manager.
Return value: Performance, a performance manager object.

Performance

.now

> This method is called using Performance.now().

Feature description: Gets the current timestamp in microseconds.
Return value: number, a timestamp.

Subpackage loading

Note:

Only supported in mini games.

preDownloadSubpackage

This method is called using wx.preDownloadSubpackage(Object object).

Feature description: Triggers subpackage pre-download.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
packageType	string	'normal'	False	Subpackage type. The name field becomes invalid if you enter "workers" for this property.
name	string		True	Subpackage name. You can enter name or root. In an independent subpackage, __GAME__ indicates loading the main package.
success	function		True	Callback function for successful subpackage loading.
fail	function		True	Callback function for failed subpackage loading.
complete	function		True	Callback function executed after the subpackage loading ends (regardless of success or failure).

Return value: PreDownloadSubpackageTask. Instance of a subpackage pre-download task, used to get the pre-download status of the subpackage.

Note: The difference between wx.preDownloadSubpackage and wx.loadSubpackage: wx.preDownloadSubpackage is only used to download the code package without automatically executing the code, and wx.loadSubpackage will automatically execute the code after downloading the code package.

Example:

Example for worker subpackage:

```
 // Configure workers as a subpackage in app.json / game.json first
{
  "workers": {
    "path": "myWorkersFolder",
    "isSubpackage": true // true means the worker is packaged as a subpackage Defau
  }
}
```

```
// Call wx.preDownloadSubpackage to download the worker subpackage. Only after the
var task = wx.preDownloadSubpackage({
  packageType: "workers",
  success(res) {
    console.log("load worker success", res)
    wx.createWorker("myWorkersFolder/request/index.js") // Create worker. If the wo
  },
  fail(res) {
    console.log("load worker fail", res)
  }
})

task.onProgressUpdate(res => {
  console.log(res.progress) // Listen to download progress using onProgressUpdate
  console.log(res.totalBytesWritten)
  console.log(res.totalBytesExpectedToWrite)
})
```

Example for ordinary subpackage:

```
  // Configure the subpackage in app.json / game.json first
{
    "subPackages": [
      {
        "name": "ModuleA",
        "root": "/ModuleA/"
      }
    ]
}
```

```
var task = wx.preDownloadSubpackage({
  name: "ModuleA",
  success(res) {
    console.log("load subpackage success", res)
    // Execute the subpackage code
    wx.loadSubpackage({
      name: "ModuleA",
      success(res) {
        console.log(res)
      },
    })
  },
  fail(res) {
    console.log("load subpackage fail", res)
  }
```

```
})

task.onProgressUpdate(res => {
  console.log(res.progress) // Listen to download progress using onProgressUpdate
  console.log(res.totalBytesWritten)
  console.log(res.totalBytesExpectedToWrite)
})
```

PreDownloadSubpackageTask

.onProgressUpdate

This method is called using PreDownloadSubpackageTask.onProgressUpdate(function listener).

Feature description: Listens for subpackage loading progress update event.
Parameter and description:function listener. The listener for subpackage loading progress update event. See the parameter Object.res as below.

Property	Type	Description
progress	number	Subpackage download progress in percentage.
totalBytesWritten	number	The downloaded data length in bytes.
totalBytesExpectedToWrite	number	The total length of data expected to be downloaded, in bytes.

loadSubpackage

This method is called using wx.loadSubpackage(Object object).

Feature description: Triggers subpackage loading.
Parameter and description: Object object.

Property	Type	Default value	Required	Description
packageType	string	'normal'	False	Subpackage type. The name field becomes invalid if you enter "workers" for this property.
name	string		True	Subpackage name. You can enter name or root. In an

				independent subpackage, __GAME__ indicates loading the main package.
success	function		True	Callback function for successful subpackage loading.
fail	function		True	Callback function for failed subpackage loading.
complete	function		True	Callback function executed after the subpackage loading ends (regardless of success or failure).

Return value: LoadSubpackageTask. Instance of a subpackage loading task, used to get the loading status of the subpackage.
Note: The difference between wx.preDownloadSubpackage and wx.loadSubpackage: wx.preDownloadSubpackage is only used to download the code package without automatically executing the code, and wx.loadSubpackage will automatically execute the code after downloading the code package.

LoadSubpackageTask

.onProgressUpdate

This method is called using LoadSubpackageTask.onProgressUpdate(function listener).

Feature description: Listens for subpackage loading progress update event.
Parameter and description:function listener. The listener for subpackage loading progress update event. See the parameter Object.res as below.

Property	Type	Description
progress	number	Subpackage download progress in percentage.
totalBytesWritten	number	The length of data that has been downloaded, in bytes.
totalBytesExpectedToWrite	number	The total length of data expected to be downloaded, in bytes.

Routing

Last updated：2024-11-21 18:41:53

switchTab

This API is used via wx.switchTab(Object object).

Feature Description: Redirects to the tabBar page, simultaneously closing all other non-tabBar pages.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
url	string	-	Yes	The path to the tabBar page that needs to be redirected (code package path) (must be defined on the page in the tabBar field of app.json). The path cannot carry parameters.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Successful Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Sample Code

```
// app.json
{
  "tabBar": {
    "list": [{
      "pagePath": "index",
      "text": "Homepage"
    },{
      "pagePath": "other",
      "text": "Other"
    }]
  }
}
```

```
wx.switchTab({
  url: '/index'
})
```

reLaunch

This method is used via wx.reLaunch(Object object).

Feature Description: Closes all pages and opens a specific page within the application.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
url	string	-	Yes	The path to a non-tabBar page within the application to which you want to redirect (code package path). This path can be followed by parameters. Parameters are separated from the path by a `?`, with parameter keys and values connected by a `=`, and different parameters separated by a `&`. For example, `'path?key=value&key2=value2'`
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Sample Code

```
wx.reLaunch({
  url: 'test?id=1'
})
```

redirectTo

This API is used via wx.redirectTo(Object object).

Feature Description: Close the current page and jump to a page within the app. However, it is not allowed to jump to a tabbar page.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
url	string	-	Yes	The path to a page within the application to which you want to redirect (code package path). This path can be followed by parameters. Parameters are separated from the path by a `?`, with parameter keys and values connected by a =, and different parameters separated by a `&`. For example, `'path?key=value&key2=value2'`
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Sample Code

```
wx.redirectTo({
  url: 'test?id=1'
})
```

navigateTo

This API is used via wx.navigateTo(Object object).

Feature Description: Retains the current page and redirects to a specific page within the application, but cannot redirect to a tabbar page. Using wx.navigateBack allows a return to the original page. The page stack in the mini program can be up to ten layers deep.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
url	string	-	Yes	The path to the non-tabBar page within the application to which you want to navigate (code package path). This path can be followed by parameters. Parameters are separated from the path by a `?`, parameter keys and values are connected by an equals sign, and different parameters are separated by a `&`; for example, `'path?key=value&key2=value2'`
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Sample Code

```
wx.navigateTo({
  url: 'test?id=1',
})


//test.js
Page({
  onLoad: (option){
      console.log(option.query)
    }
})
```

navigateBack

This API is used via wx.navigateBack(Object object).

Feature Description: Close the current page, return to the previous page or multi-level page, you can get the current page stack by getCurrentPages, decide how many levels you need to return.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
delta	number	1	No	The number of pages returned. If delta exceeds the number of existing pages, it will return to the home page.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Sample Code

```
// Note: When calling navigateTo for navigation, the page that calls this method wi

// This is page A
wx.navigateTo({
  url: 'B?id=1'
})

// This is page B
wx.navigateTo({
  url: 'C?id=1'
})

// Perform navigateback within page C will return to page A.
wx.navigateBack({
  delta: 2
})
```

EventChannel

Inter-page event communication channel.

EventChannel.emit

This method is used via EventChannel.emit(string eventName, any args).

Feature Description: Triggers an event.
Parameter and Description: string eventName, event name; any args, event parameter.

EventChannel.off

This method is used via EventChannel.off(string eventName, EventCallback fn).

Feature Description: Cancels the monitoring of an event. When a second parameter is provided, only the specified monitoring function is canceled; otherwise, all monitoring functions are canceled. The method is used via EventChannel.off(string eventName, function fn).
Parameter and Description: string eventName, event name; function fn, event listener function; any args, parameters of triggered event.

EventChannel.on

This method is used via EventChannel.on(string eventName, EventCallback fn).

Feature Description: Continuously monitors an event. This method is used via EventChannel.on(string eventName, function fn).
Parameter and Description: string eventName, event name; function fn, event listener function; any args, parameters of triggered event.

EventChannel.once

This method is used via EventChannel.once(string eventName, EventCallback fn)

Feature Description: Monitors an event once, becoming invalid after triggering. This method is used via EventChannel.once(string eventName, function fn).
Parameter and Description: string eventName, event name; function fn, event listener function; any args, parameters of triggered event.

©2013-2025 Tencent Cloud International Pte. Ltd.

Navigation

Last updated：2025-02-25 14:19:31

exitMiniProgram

This API is called using wx.exitMiniProgram(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Exits the current mini program. This API can only be successfully called as a result of a user tap action.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

navigateToMiniProgram

This API is called using wx.navigateToMiniProgram(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Opens another mini program.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
appId	string	-	True	The appId of the mini program to open.

path	string	-	False	The page path to open. If empty, the home page is opened. The string after the ? in the path will become a query, and the query data can be obtained in the mini program's callback functions App.onLaunch, App.onShow and Page.onLoad, or in wx.getLaunchOptionsSync.
extraData	object	-	False	Data to pass to the target mini program. It can be obtained in `App.onLaunch` and `App.onShow` .
envVersion	string	release	False	The version of the mini program to open. This parameter is valid only when the current mini program is in development version or Preview. If the current mini program is the released version, the mini program that is opened must be the released version. Valid values: develop: The development version trial: The Preview release: The released version
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Usage restrictions

User trigger required

If the user does not tap anywhere on the mini program page, the developer will not be able to call this API to automatically navigate to another mini program.

User confirmation required

Before navigating to another mini program, a pop-up window will appear asking for user confirmation, and the navigation will proceed after the user confirms that. If the user taps**Cancel**, it will trigger a callback with `fail cancel` .

Each mini program can jump to no more than 10 other mini programs

When submitting a new version of mini program code, developers must include a list of up to 10 target mini programs in the code configuration if using the navigation feature. Otherwise, the code will not be approved. The list can be updated with each new version but cannot be dynamically modified. For more information about the configuring

method, see Global Configuration. The appId must be in the configuration list, otherwise the fail callback will be triggered with `fail appId "${appId}" is not in navigateToMiniProgramAppIdList`.

Example:

```
wx.navigateToMiniProgram({
  appId: '',
  path: 'page/index/index?id=123',
  extraData: {
    foo: 'bar'
  },
  envVersion: 'develop',
  success(res) {
    // Navigation to mini program successful
  }
})
```

navigateBackMiniProgram

This API is called using wx.navigateBackMiniProgram(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Returns to the previous mini program. This API can be called successfully only when the current mini program is opened by another mini program.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
extraData	Object	{}	False	Data to return to the previous mini program. The previous mini program can access this data in App.onShow.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
wx.navigateBackMiniProgram({
  extraData: {
    foo: 'bar'
  },
  success(res) {
    // Returned to previous mini program successfully
  }
})
```

Share

Last updated：2025-05-26 10:49:06

hideShareMenu

This API is called using wx.hideShareMenu (Object object).

Feature description: Hides the “Send to Chat” button when the top-right menu is tapped.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
hideShareItems	Array.<string>	['wechatFriends', 'wechatMoment']	False	wechatFriends: Controls whether to hide “Send to Chat” 'wechatMoment: Controls whether to hide the "Share on Moments" option. If the hideShareItems parameter is not provided, the default is to hide all options: "Share" and "Send to Chat".
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
wx.hideShareMenu({
    hideShareItems: ['wechatFriends', 'wechatMoment']
})
```

showShareMenu

This API is called using wx.showShareMenu (Object object).

©2013-2025 Tencent Cloud International Pte. Ltd.

Feature description: Displays the "Share", "Send to Chat", and "Share on Moments" buttons in the top-right menu.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
showShareItems	Array.<string>	['wechatFriends', 'wechatMoment']	False	Please check below object.showShareItems Parameter and description:
withShareTicket	boolean	false	False	Whether to use shareTicket for sharing.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.showShareItems parameters

Value	Description
'wechatFriends'	Controls whether to hide “Send to Chat”
'wechatMoment'	Controls whether to hide the "Share on Moments" option. If the hideShareItems parameter is not provided, the default is to hide all options: "Share" and "Send to Chat".

Example:

```
wx.showShareMenu({
  showShareItems: ['wechatFriends', 'wechatMoment']
})
```

updateShareMenu

This API is called using wx.updateShareMenu (Object object).

Feature description: Updates the sharing property.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
withShareTicket	boolean	false	False	Whether to use shareTicket for sharing.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
wx.updateShareMenu({
  withShareTicket: true
})
```

showShareImageMenu

This API is called using wx.showShareImageMenu (Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Opens a share image dialog to send, save, or download an image.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
path	string	-	True	The local or temporary path of the image to share.
style	string	default	False	The style of sharing.

needShowEntrance	string	false	False	Whether the shared image message should include a mini program entrance. (Only some mini program categories are available)
entrance	string	false	False	The path to open the mini program from the message entrance. If the current page allows sharing with friends, it defaults to the current page path; otherwise, it defaults to the mini program's home page.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
wx.downloadFile({
  url: 'https://res.wx.qq.com/wxdoc/dist/assets/img/demo.ef5c5bef.jpg',
  success: (res) => {
    wx.showShareImageMenu({
      path: res.tempFilePath
    })
  }
})
```

onCopyUrl

This API is called using wx.onCopyUrl(function listener).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Listens for the event triggered when the user taps the "Copy URL" button in the top-right menu.

This API is in beta and currently only supported on the Android platform.

Parameter and description:

The function that listens for event triggered when the user taps the "Copy URL" button in the top-right menu.

Object res,

Property	Type	Description

query	string	The query string carried by the current page when opened with a short link. When using this in a mini program, you should call wx.onCopyUrl to customize the query when entering the page and call wx.offCopyUrl when exiting the page to prevent affecting other pages.

Example:

```
// Bind the shared parameters
wx.onCopyUrl(() => {
  return { query: 'a=1&b=2' }
})

// Unbind the shared parameters
wx.offCopyUrl()
```

offCopyUrl

This API is called using wx.offCopyUrl().

Note:

This API is supported in mini programs but not in mini games.

Feature description: Removes all listeners for event triggered when the user taps the "Copy URL" button in the top-right menu.

Example:

```
// Bind the shared parameters
wx.onCopyUrl(() => {
  return { query: 'a=1&b=2' }
})

// Unbind the shared parameters
wx.offCopyUrl()
```

©2013-2025 Tencent Cloud International Pte. Ltd.

©2013-2025 Tencent Cloud International Pte. Ltd.

Interface
Window

Last updated：2025-02-24 17:10:35

onWindowResize

This API is called using wx.onWindowResize(function listener).

Notes:

This API is supported in mini programs but not in mini games.

Feature description: Listens for window size change event.

Parameter and description:function listener. listener for window size change event.

Property	Type
size	Object

size object property

Property	Type	Description
windowWidth	number	The new window width in pixels.
windowHeight	number	The new window height in pixels.

offWindowResize

The API is called using wx.offWindowResize(function listener).

Notes:

This API is supported in mini programs but not in mini games.

Feature description: Removes the listener for the window size change event.

Parameter and description:function listener. The listener that was passed to onWindowResize. If not provided, all listeners will be removed.

©2013-2025 Tencent Cloud International Pte. Ltd.

```
const listener = function (res) { console.log(res) }

wx.onWindowResize(listener)
wx.offWindowResize(listener) // Must pass the same function object used in wx.onWin
```

©2013-2025 Tencent Cloud International Pte. Ltd.

Font

Last updated：2025-01-23 17:55:46

loadFontFace

This API is used via wx.loadFontFace(Object object).

Feature Description: Dynamically loads web fonts. The file address must be of download type, and it needs to be invoked within js.

Note:

The contet-type returned by the font file refers to the font, and will fail to parse if it is not in the correct format;
For true machine compatibility, the font link must be https;
The link to the font must be under the same source, or enable cors support, the domain name of the mini program is servicewechat.com(If the client has a custom domain name, please adjust it here according to the custom configuration).;
The tooltip Faild to load font can be ignored.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
family	string	-	Yes	Defined font name
source	string	-	Yes	The address of the font resource. The recommended formats are TTF and WOFF, as WOFF2 may not be compatible with older versions of iOS.
desc	object	-	No	Optional font descriptors
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Successful Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Structural value of 'desc'

Structure	Type	Default	Required	Description

attributes		value		
style	string	normal	No	Font style, with optional values being 'normal', 'italic', or 'oblique'.
weight	string	normal	No	Font weight, with optional values being 'normal', 'bold', '100', '200', or '900'.
variant	string	normal	No	Sets the display text for small capital letters, with optional values being 'normal', 'small-caps', or 'inherit'.

Parameters for object.success callback function: Object res.

Attribute	Type	Description
status	string	Font Loading Results

Parameters for object.fail callback function: Object res

Attribute	Type	Description
status	string	Font Loading Results

object.complete Callback Function Parameters: Object res

Attribute	Type	Description
status	string	Font Loading Results

Sample Code

```
// https://sungd.github.io/Pacifico.ttf Font links need to have https enabled and c
wx.loadFontFace({
  family: ‘Bitstream Vera Serif Bold’,
  family: ‘MyFont’.
  source: ‘url(’https://sungd.github.io/Pacifico.ttf‘)’,
  Success: console.log
})
```

```
page {
  font-family: 'MyFont';
}
```

Interaction

Last updated：2025-02-24 17:10:35

hideLoading

This API is called using wx.hideLoading(Object object).

Feature description: Hides the loading prompt box.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
noConflict	boolean	false	False	Currently, toast and loading related APIs can be used interchangeably. This parameter can be used to cancel this feature.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

hideToast

This API is called using wx.hideToast(Object object).

Feature description: Hides the message prompt box.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
noConflict	boolean	false	False	Currently, toast and loading related APIs can be used interchangeably. This parameter can be used to cancel this feature.

success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

showActionSheet

This API is called using wx.showActionSheet(Object object).

Notes:

For Android versions earlier than 6.7.2, tapping cancel or the mask will trigger the fail callback with errMsg as "fail cancel".

For Android 6.7.2 and later, and iOS, tapping the mask will not close the modal popup. In the fail callback function, if errMsg is "fail cancel", it is not recommended to implement business logic in this scenario.

Feature description: Shows the action menu.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
itemList	array.<string>	-	True	Array of button text, with a maximum length of 6.
itemColor	string	#000000	False	Button text color.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.success callback function parameter: Object res

Property	Type	Description
tapIndex	number	The index of the button tapped by the user, starting from 0.

Example:

```
wx.showActionSheet({
  itemList: ['A', 'B', 'C'],
  success (res) {
    console.log(res.tapIndex)
  },
  fail (res) {
    console.log(res.errMsg)
  }
})
```

showLoading

This API is called using wx.showLoading(Object object).

Feature description: Shows the loading prompt box. You need to call wx.hideLoading to close the prompt box.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
title	string	-	True	The content of the prompt.
mask	boolean	false	False	Whether to show a transparent mask to prevent touch penetration.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
wx.showLoading({
  title: 'Loading',
})

setTimeout(function () {
  wx.hideLoading()
}, 2000)
```

Notes:

wx.showLoading and wx.showToast cannot be shown at the same time.
wx.showLoading should be used with wx.hideLoading.

showModal

This API is called using is wx.showModal(Object object).

Notes:

For Android versions earlier than 6.7.2, tapping cancel or the mask will trigger the fail callback with errMsg as "fail cancel".
For Android 6.7.2 and later, and iOS, tapping the mask will not close the modal popup. In the fail callback function, if errMsg is "fail cancel", it is not recommended to implement business logic in this scenario.

Feature description: Shows the modal dialog box.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
title	string	-	False	The title of the prompt.
content	string	-	False	The content of the prompt.
showCancel	boolean	true	False	Whether to show the Cancel button.
cancelText	string	Cancel	False	The text of the Cancel button, which can contain up to four characters.
cancelColor	string	#000000	False	The color of the Cancel button text, must be a hexadecimal color string.
confirmText	string	OK	False	The text of the OK button, which can contain up to four characters.
confirmColor	string	#576B95	False	The color of the OK button text, must be a hexadecimal color string.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

©2013-2025 Tencent Cloud International Pte. Ltd.

object.success callback function parameter: Object res

Property	Type	Description
confirm	boolean	true if the user tapped the OK button.
cancel	boolean	true if the user tapped the Cancel button (used to distinguish between tapping the mask to close or tapping the Cancel button on Android).

Example:

```
wx.showModal({
  title: 'Prompt',
  content: 'This is a modal pop-up',
  success (res) {
    if (res.confirm) {
      console.log('The user taps OK')
    } else if (res.cancel) {
      console.log('The user taps Cancel')
    }
  }
})
```

showToast

This API is called using wx.showToast(Object object).

Feature description: Shows the message prompt box.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
title	string	-	True	The content of the prompt.
icon	string	success	False	Icon. Valid values: success: Shows a success icon, with a maximum of seven characters for the title. error: Shows a failure icon, with a maximum of seven characters for the title. loading: Shows a loading icon, with a maximum of seven characters for the title.

				none: No icon is displayed, and the title can display up to two lines.
image	string	-	False	The local path to a custom icon. The image has a higher priority than icon.
duration	number	1500	False	The duration of the prompt in milliseconds.
mask	boolean	false	False	Whether to show a transparent mask to prevent touch penetration.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
wx.showToast({
  title: 'Success',
  icon: 'success',
  duration: 2000
})
```

Notes:

wx.showLoading and wx.showToast cannot be shown at the same time.

wx.showLoading should be used with wx.hideLoading.

enableAlertBeforeUnload

This API is called using wx.enableAlertBeforeUnload(Object object).

Notes:

This API is supported in mini programs but not in mini games.

Feature description: Enables a confirmation dialog when the user tries to navigate away from a page in the mini program.

Conditions for showing the dialog box:

When the user is on a non-home page or the bottom-most page in the mini program.

When using the official navigation bar's back button.

When using a custom back button in full-screen mode.

When pressing the back button on an Android device.

Notes:

This feature does not intercept swipe gestures for navigation.

In any scenario, this feature should not prevent the user from exiting the mini program.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
message	string	-	True	The content of the confirmation dialog.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

disableAlertBeforeUnload

This API is called using wx.disableAlertBeforeUnload(Object object).

Notes:

This API is supported in mini programs but not in mini games.

Feature description: Disables the confirmation dialog when the user tries to navigate away from a page in the mini program.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Navigation Bar

Last updated：2024-11-21 18:41:53

setNavigationBarTitle

This API is used via wx.setNavigationBarTitle(Object object).

Feature Description: Dynamically sets the title of the current page.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
title	string	-	Yes	Page title
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

```
wx.setNavigationBarTitle({
  title: 'Current Page'
})
```

setNavigationBarColor

This API is used via wx.setNavigationBarColor(Object object).

Feature Description: Set the color scheme for the page navigation bar.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description

frontColor	string	-	Yes	Foreground color value, including the color of buttons, title, and status bar, only supporting #ffffff and #000000.
backgroundColor	string	-	Yes	Background color value, with valid values being hexadecimal colors.
animation	object	-	No	Animation Effect
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Structure attributes of object.animation

Attribute	Type	Default value	Required	Description
duration	number	0	No	Animation transition duration, measured in milliseconds.
timingFunc	string	'linear'	No	Animation transition method

Valid values for object.animation.timingFunc

Value	Description
'linear'	The animation maintains a consistent speed from beginning to end.
'easeIn'	The animation starts at a slow pace.
'easeOut'	The animation concludes at a slow pace.
'easeInOut'	The animation starts and concludes at a slow pace.

Sample Code

```
wx.setNavigationBarColor({
  frontColor: '#ffffff',
  backgroundColor: '#ff0000',
  animation: {
    duration: 400,
    timingFunc: 'easeIn'
```

```
    }
})
```

hideHomeButton

This API is used via wx.hideHomeButton(Object object).

Feature Description: Hide home button. When the bottom page of the mini program opened by the user is not the home page, the sharing panel will show the "Back to Home" button by default, and the developer can call hideHomeButton in the onShow of the page to hide it.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

showNavigationBarLoading

This API is used via wx.showNavigationBarLoading(Object object).

Feature Description: Displays a navigation bar loading animation on the current page.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the

				interface invocation (both successful and unsuccessful invocations)

hideNavigationBarLoading

This API is used via wx.hideNavigationBarLoading(Object object).

Feature Description: Hides the navigation bar loading animation on the current page.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

©2013-2025 Tencent Cloud International Pte. Ltd.

Background

Last updated：2025-04-03 17:53:13

setBackgroundColor

This API is used via wx.setBackgroundColor(Object object).

Feature Description: Dynamically sets the background color of the window. Please add "enablePullDownRefresh":true to the page configuration file for testing.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Note
backgroundColor	string	-	No	Background color value, which must be a hexadecimal color value.
backgroundColorTop	string	-	No	Background color of the top window, which must be a hexadecimal color value. This feature is supported only on iOS.
backgroundColorBottom	string	-	No	Background color of the bottom window, which must be a hexadecimal color value. This feature is supported only on iOS.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Sample Code

```
wx.setBackgroundColor({
  backgroundColor: '#ffffff', // The background color of the window is white.
})

wx.setBackgroundColor({
```

```
  backgroundColorTop: '#ffffff', // The background color of the top window is white
  backgroundColorBottom: '#ffffff', // The background color of the bottom window is
})
```

setBackgroundTextStyle

This API is used via wx.setBackgroundTextStyle(Object object).

Feature Description: Dynamically sets the style of the dropdown background font and loading image. Please add "enablePullDownRefresh":true to the page configuration file for testing.

Parameter and Description: Object

Attribute	Type	Default value	Required	Note
textStyle	string	-	Yes	The style of the dropdown background text and loading image.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Valid values of object.textStyle

Value	Note
dark	Dark mode
light	Light mode

Sample Code

```
wx.setBackgroundTextStyle({
    textStyle: 'dark' // The dropdown background text and loading icon styles are s
})
```

TabBar

Last updated：2025-03-20 18:01:56

showTabBar

The method of utilizing this API is wx.showTabBar(Object object).

Function Description: Unveils the tabBar.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
animation	boolean	false	Not required	Is animation effect necessary?
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

hideTabBar

The usage method for this API is wx.hideTabBar(Object object)

Function Description: Conceals the tabBar.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
animation	boolean	false	Not	Is animation effect necessary?

©2013-2025 Tencent Cloud International Pte. Ltd.

			required	
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

setTabBarStyle

The usage method for this API is wx.setTabBarStyle(Object object)

Function Description: Dynamically configures the overall style of the tabBar.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
color	string	-	Not required	Default text color on the tab, HexColor
selectedColor	string	-	Not required	Text color on the tab when selected, HexColor
backgroundColor	string	-	Not required	Background color of the tab, HexColor
borderStyle	string	-	Not required	Color of the border on the tab, only supports black/white
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

setTabBarItem

The usage method for this API is wx.setTabBarItem(Object object)

Function Description: Dynamically sets the content of a specific tabBar item.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
index	number	-	Supported	Which item of the tabBar, counting from the left
text	string	-	Not required	Text on the tab button
iconPath	string	-	Not required	Image path, the icon size is limited to 40kb, with a recommended dimension of 81px * 81px. This parameter is invalid when the position is set to 'top'.
selectedIconPath	string	-	Not required	Path of the image when selected, the icon size is limited to 40kb, with a recommended dimension of 81px * 81px. This parameter is invalid when the position is set to 'top'.
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Example Code

```
wx.setTabBarItem({
  index: 0,
  text: 'text',
  iconPath: '/path/to/iconPath',
  selectedIconPath: '/path/to/selectedIconPath'
})
```

©2013-2025 Tencent Cloud International Pte. Ltd.

showTabBarRedDot

The usage of this API is wx.showTabBarRedDot(Object object)

Function Description: Display the red dot at the top right corner of a tabBar item.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
index	number	-	Supported	Which item of the tabBar, counting from the left
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

hideTabBarRedDot

The usage of this API is wx.hideTabBarRedDot(Object object)

Function Description: Hide the red dot at the top right corner of a tabBar item.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
index	number	-	Supported	Which item of the tabBar, counting from the left
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation

complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

setTabBarBadge

The usage of this API is wx.setTabBarBadge(Object object)

Function Description: Add text to the top right corner of a tabBar item.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
index	number	-	Supported	Which item of the tabBar, counting from the left
text	string	-	Supported	The displayed text truncates to '...' if it exceeds 4 characters.
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Example Code

```
wx.setTabBarBadge({
  index: 0,
  text: '1'
})
```

removeTabBarBadge

The usage of this API is wx.removeTabBarBadge(Object object)

Function Description: Remove the text from the top right corner of a tabBar item.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
index	number	-	Supported	Which item of the tabBar, counting from the left
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

©2013-2025 Tencent Cloud International Pte. Ltd.

Pull-down refresh

Last updated：2025-04-03 17:53:13

startPullDownRefresh

This API is used via wx.startPullDownRefresh(Object object).

Feature Description: Starts pull-down refresh. Upon invocation, it triggers the pull-down refresh animation, mirroring the effect of a user-initiated pull-down refresh. Please add "enablePullDownRefresh":true to the page configuration file.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Sample Code

```
wx.startPullDownRefresh()
```

stopPullDownRefresh

This API is used via wx.stopPullDownRefresh(Object object).

Feature Description: Disables pull-down refresh on the current page. Please add "enablePullDownRefresh":true to the page configuration file.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description

success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Sample Code

```
Page({
  onPullDownRefresh () {
    wx.stopPullDownRefresh()
  }
})
```

Scroll

Last updated：2025-04-28 18:13:10

pageScrollTo

This API is used via wx.pageScrollTo(Object object).

Feature Description: Scrolls the page to the desired location.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
scrollTop	number	-	Yes	Scroll to the target position on the page, measured in pixels.
duration	number	300	No	Duration of the scrolling animation, measured in milliseconds.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Sample Code

```
wx.pageScrollTo({
  scrollTop: 0,
  duration: 300
})
```

ScrollViewContext

Feature Description: Enhances ScrollView instance, which can be obtained through the wx.createSelectorQuery's NodesRef.node method. Effective only when the scroll-view component's enhanced attribute is enabled.

Attribute and Descriptions

Attribute	Type	Description
scrollEnabled	boolean	Scrolling Switch
bounces	boolean	Set the scroll boundary elasticity (currently unsupported)
showScrollbar	boolean	Set the visibility of the scrollbar
pagingEnabled	boolean	Pagination Slide Switch
fastDeceleration	boolean	Set scroll deceleration rate (iOS only)
decelerationDisabled	boolean	Cancel the scroll inertia (Effective only on iOS)

Sample Code

```
wx.createSelectorQuery()
  .select('#scrollview')
  .node()
  .exec((res) => {
    const scrollView = res[0].node;
    scrollView.scrollEnabled = false;
  })
```

.scrollIntoView

The method is used via: ScrollViewContext.scrollIntoView(string selector, object ScrollIntoViewOptions)

Feature Description: Scrolls the element to the top or bottom of the viewport.

Parameters and Description: string selector, element selector.

.scrollTo

The method is used via ScrollViewContext.scrollTo(Object object)

Feature Description: Scroll the element to the specified position, accurate to pixels.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
top	number	-	No	Top distance
left	number	-	No	Left Boundary Distance
velocity	number	-	No	Initial Speed (iOS only)
duration	number	-	No	Scroll animation length (iOS only)
animated	boolean	-	No	Enable scroll animation or not

©2013-2025 Tencent Cloud International Pte. Ltd.

Animation

Last updated：2025-04-07 17:35:14

createAnimation

This API is used via wx.createAnimation(Object object).

Feature Description: Creates an animation instance animation and invokes the instance's methods to depict the animation, before ultimately exporting the animation data through the instance's export method and passing it to the component's animation attribute.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Note
duration	number	400	No	Animation duration, measured in milliseconds.
timingFunction	string	'linear'	No	Animation effect
delay	number	0	No	Animation delay duration, measured in milliseconds.
transformOrigin	string	'50% 50% 0'	No	-

Valid values of timingFunction

Value	Note
'linear'	The animation maintains a consistent speed from beginning to end.
'ease'	The animation starts at a slow pace, then accelerates, and finally decelerates before concluding.
'ease-in'	The animation starts at a slow pace.
'ease-in-out'	The animation starts and concludes at a slow pace.
'ease-out'	The animation concludes at a slow pace.
'step-start'	The animation leaps to its concluding state from the very first frame and maintains this until the end.

©2013-2025 Tencent Cloud International Pte. Ltd.

'step-end'	The animation consistently maintains its initial state, leaping to its concluding state only in the final frame.

Return Value：Animation.

Animation

Animation object.

Sample code

Corresponding WXML file

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

Corresponding js file

```
Page({
  data: {
    animationData: {}
  },
  onShow() {
    const 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() {
    // Simultaneous rotation and enlargement
    this.animation.rotate(45).scale(2, 2).step()
    this.setData({
      animationData: this.animation.export()
    })
  },
  rotateThenScale() {
    // Rotate first, then enlarge.
    this.animation.rotate(45).step()
    this.animation.scale(2, 2).step()
    this.setData({
      animationData: this.animation.export()
    })
  },
  rotateAndScaleThenTranslate() {
    // Simultaneous rotation and enlargement before translation
    this.animation.rotate(45).scale(2, 2).step()
    this.animation.translate(100, 100).step({duration: 1000})
    this.setData({
      animationData: this.animation.export()
    })
  }
})
```

.backgroundColor

The method is used via Animation.Animation.backgroundColor(string value).

Feature Description: Sets the background color.

Parameter and Description: string value, color value.

Return Value：Animation.

.bottom

This method is used via Animation.Animation.bottom(number|string value).

Feature Description: Sets the bottom value.

Parameter and Description: number|string value, length value. If a number is passed in, px is used by default. Length values in other custom units can also be passed in.

©2013-2025 Tencent Cloud International Pte. Ltd.

Return Value：Animation.

.export

This API is used via Object Animation.export().

Feature Description: Exports the animation queue. The export method will clear the previous animation operations each time it is called.
Return Value: Object, animationData.

Attribute	Type
actions	Array.<Object>

.height

This method is used via Animation Animation.height(number|string value).

Feature Description: Sets height.
Parameter and Description: number|string value, length value. If a number is passed in, px is used by default. Length values in other custom units can also be passed in.
Return Value：Animation.

.left

This method is used via Animation Animation.left(number|string value).

Feature Description: Sets the left value.
Parameter and Description: number|string value, length value. If a number is passed in, px is used by default. Length values in other custom units can also be passed in.
Return Value：Animation.

.matrix

This API is used via Animation Animation.matrix().

Feature Description: Consistent with transform-function matrix.
Return Value：Animation.

.matrix3d

This API is used via Animation Animation.matrix3d()

Feature Description: Consistent with transform-function matrix.
Return Value：Animation.

.opacity

This method is used via Animation Animation.opacity(number value).

Feature Description: Sets the opacity.
Parameter and Description: number value, opacity, ranging from 0 to 1.
Return Value：Animation.

.right

This method is used via Animation Animation.right(number|string value).

Feature Description: Sets the right value.
Parameter and Description: number|string value, length value. If a number is passed in, px is used by default. Length values in other custom units can also be passed in.
Return Value：Animation.

.rotate

This API is used via Animation Animation.rotate(number angle).

Feature Description: Rotate one angle clockwise from the origin.
Parameter and Description: number angle, the degree of rotation. Range [-180, 180].
Return Value：Animation.

©2013-2025 Tencent Cloud International Pte. Ltd.

.rotate3d

This API is used via Animation Animation.rotate3d(number x, number y, number z, number angle).

Feature Description: Rotates a degree clockwise from a fixed axis.
Parameter and Description: number x, the x-coordinate of the rotation axis; number y, the y-coordinate of the rotation axis; number z, the z-coordinate of the rotation axis; number angle, the degree of rotation. Range [-180, 180].
Return Value：Animation.

.rotateX

This API is used via Animation Animation.rotateX(number angle).

Feature Description: Rotates a degree clockwise from the X-axis.
Parameter and Description: number angle, the degree of rotation. Range [-180, 180].
Return Value：Animation.

.rotateY

This method is used via Animation Animation.rotateY(number angle).

Feature Description: Rotates a degree clockwise from the Y-axis.
Parameter and Description: number angle, the degree of rotation. Range [-180, 180].
Return Value：Animation.

.rotateZ

This Method is used via Animation Animation.rotateZ(number angle).

Feature Description: Rotates a degree clockwise from the Z-axis.
Parameter and Description: number angle, the degree of rotation. Range [-180, 180].
Return Value：Animation.

.scale

©2013-2025 Tencent Cloud International Pte. Ltd.

This method is used via Animation Animation.scale(number sx, number sy).

Feature Description: Scaling.
Parameter and Description: number sx, when only sx parameter is present, it indicates scaling on the X and Y axes simultaneously by sx times; number sy, scaling on the Y axis by sy times.
Return Value：Animation.

.scale3d

This method is used via Animation Animation.scale3d(number sx, number sy, number sz).

Feature Description: Scaling.
Parameter and Description: number sx, the scaling factor for the x-axis; number sy, the scaling factor for the y-axis; number sz, the scaling factor for the z-axis.
Return Value：Animation.

.scaleX

This method is used via Animation Animation.scaleX(number scale).

Feature Description: Scaling on the X-axis.
Parameter and Description: number scale, the scaling factor on the X-axis.
Return Value：Animation.

scaleY

This method is used via Animation Animation.scaleY(number scale).

Feature Description: Scaling on the Y-axis.
Parameter and Description: number scale, the scaling factor on the Y-axis.
Return Value：Animation.

.scaleZ

This method is used via Animation Animation.scaleZ(number scale).

Feature Description: Scaling on the Z-axis.
Parameter and Description: number scale, the scaling factor on the Z-axis.
Return Value：Animation.

.skew

This method is used via Animation Animation.skew(number ax, number ay).

Feature Description: Skews the coordinates on the X and Y axes.
Parameter and Description: number ax, tilt angle to X-axis coordinates, range [-180, 180]; number ay, tilt angle to Y-axis coordinates, range [-180, 180].
Return Value：Animation.

.skewX

This Method is used via Animation Animation.skewX(number angle).

Feature Description: Skews the coordinates on the X-axis.
Parameter and Description: number angle, tilt angle, range [-180, 180].
Return Value：Animation.

.skewY

This method is used via Animation Animation.skewY(number angle).

Feature Description: Skews the coordinates on the Y-axis.
Parameter and Description: number angle, tilt angle, range [-180, 180].
Return Value：Animation.

.step

This API is used via Animation Animation.step(Object object).

©2013-2025 Tencent Cloud International Pte. Ltd.

Feature Description: Indicates the completion of a set of animations. Any number of animation methods can be called within a set of animations. All animations in a set will start simultaneously, and the next set of animations will only proceed after the completion of the current set.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Note
duration	number	400	No	Animation duration, measured in milliseconds.
timingFunction	string	'linear'	No	Animation effect
delay	number	0	No	Animation delay duration, measured in milliseconds.
transformOrigin	string	'50% 50% 0'	No	-

Valid values of timingFunction

Value	Note
'linear'	The animation maintains a consistent speed from beginning to end.
'ease'	The animation starts at a slow pace, then accelerates, and finally decelerates before concluding.
'ease-in'	The animation starts at a slow pace.
'ease-in-out'	The animation starts and concludes at a slow pace.
'ease-out'	The animation concludes at a slow pace.
'step-start'	The animation leaps to its concluding state from the very first frame and maintains this until the end.
'step-end'	The animation consistently maintains its initial state, leaping to its concluding state only in the final frame.

Return Value：Animation.

.top

This method is used via Animation Animation.top(number|string value).

Feature Description: Sets the top value.
Parameter and Description: number|string value, length value. If a number is passed in, px is used by default. Length values in other custom units can also be passed in.
Return Value：Animation.

.translate

This method is used via Animation Animation.translate(number tx, number ty).

Feature Description: Translation and transformation.
Parameter and Description: number tx, when this is the only parameter, it represents the offset on the X-axis by tx, measured in pixels; number ty, the distance of translation on the Y-axis, measured in pixels.
Return Value：Animation.

.translate3d

This method is used via Animation Animation.translate3d(number tx, number ty, number tz).

Feature Description: Performs a translation transformation on the xyz coordinates.
Parameter and Description: number tx, when only this parameter is present, indicates an offset tx in the X-axis in px; number ty, the distance in the Y-axis in px; number tz, the distance in the Z-axis in px.
Return Value：Animation.

.translateX

This method is used via Animation Animation.translateX(number translation).

Feature Description: Performs a translation on the X-axis.
Parameter and Description: number translation, the distance of translation on the X-axis, measured in pixels.
Return Value：Animation.

.translateY

This method is used via Animation Animation.translateY(number translation).

Feature Description: Performs a translation on the Y-axis.
Parameter and Description: number translation, the distance of translation on the Y-axis, measured in pixels.
Return Value：Animation.

.translateZ

This method is used via Animation Animation.translateZ(number translation).

Feature Description: Performs a translation on the Z-axis.
Parameter and Description: number translation, the distance of translation on the Z-axis, measured in pixels.
Return Value：Animation.

.width

This method is used via Animation Animation.width(number|string value).

Feature Description: Sets the width.
Parameter and Description: number|string value, length value. If a number is passed in, px is used by default. Length values in other custom units can also be passed in.
Return Value：Animation.

Custom component

Last updated：2024-11-21 18:41:53

nextTick

The usage method of this API is wx.nextTick(function callback).

Feature Description: Defers certain operations to be executed in the next time slice, similar to the setTimeout feature.

Parameter and Description: The function callback is used because the setData and triggerEvent interfaces in custom components are inherently synchronous operations. When these interfaces are called consecutively, they are executed within a single synchronous process, which may lead to errors in the case of improper logic.

An extreme case: If the parent component's setData triggers the child component's triggerEvent, causing the parent component to perform setData again, and in the process, the child component is unloaded through the wx:if statement, this could potentially cause an error. Therefore, for logic that does not need to be completed within a single synchronous process, this interface can be used to defer execution to the next time slice.

Sample Code

```
Component({
  doSth() {
    this.setData({ number: 1 }) // Executed directly within the current synchronous

    wx.nextTick(() => {
      this.setData({ number: 3 }) // Executed after the current synchronous process
    })

    this.setData({ number: 2 }) // Executed directly within the current synchronous
  }
})
```

Menu

Last updated：2024-11-21 18:41:53

getMenuButtonBoundingClientRect

This API is used via wx.getMenuButtonBoundingClientRect().

Feature Description: Gets information about the layout position of the menu button (capsule button in the upper right corner). The coordinate information has the upper left corner of the screen as the origin .

Return Value: Object.

Layout Position Information of the Menu Buttons:

Attribute	Type	Description
width	number	Width, measured in pixels (px)
height	number	Height, measured in pixels (px)
top	number	Upper Boundary Coordinate, Unit: px
right	number	Right Boundary Coordinate, Unit: px
bottom	number	Lower Boundary Coordinate, Unit: px
left	number	Left Boundary Coordinate, Unit: px

©2013-2025 Tencent Cloud International Pte. Ltd.

Network
UDP Communication

Last updated：2025-03-20 18:01:56

createUDPSocket

This API is used via UDPSocket?wx.createUDPSocket().

Feature description: Creates a UDP Socket instance.
Return Value: UDPSocket, a UDP Socket instance.

UDPSocket

Note:
A UDP Socket instance, using IPv4 protocol by default.

.bind

This method is used via number UDPSocket.bind(number port).

Feature Description: Binds to a system-assigned random available port, or bind to a specified port number.
Parameter and Description: number port, No. of specified port to be bound. If this parameter is not passed in, the system randomly assigns an available port.
Return Value: number, the successfully bound port number.
Sample Code

```
const udp = wx.createUDPSocket()
const port = udp.bind()
```

.close

This method is used via UDPSocket.close().

Feature Description: Closes the UDP Socket instance, effectively terminating it. After closure, the UDP Socket instance can no longer send messages. Each call to UDPSocket.send will trigger an error event, and the message event callback function will no longer be executed. After a UDPSocket instance is created, it is strongly referenced by Native to ensure it is not garbage collected. After UDPSocket.close, the strong reference is removed, allowing the UDPSocket instance to be subject to garbage collection.

.connect

This method is used via UDPSocket.connect(Object object).

Feature Description: Pre-connects to the specified IP and port, to be used in conjunction with the write method.
Parameter and Description: Object

Attribute	Type	Default value	Required	Note
address	string	-	Yes	Destination for message transmission
Port No.	number	-	Yes	Port number for message transmission

Sample Code

```
const udp = wx.createUDPSocket()
udp.bind()
udp.connect({
  address: '192.168.193.2',
  port: 8848,
})
udp.write({
  address: '192.168.193.2',
  port: 8848,
  message: 'hello, how are you'
})
```

.onClose

This method is used via UDPSocket.onClose(function listener).

Feature Description: Monitors closure events.
Parameter and Description: The listener function of closure events.

.offClose

This method is used via UDPSocket.offClose(function listener).

Feature Description: Removes the listener function for closure events.
Parameter and Description: function listener, the listener function passed in by onClose. If this parameter is not passed in, all listener functions will be removed.
Sample Code

```
const listener = function (res) { console.log(res) }

UDPSocket.onClose(listener)
UDPSocket.offClose(listener) // The same function object as the listener must be pa
```

.onError

This method is used via UDPSocket.onError(function listener).

Feature Description: Monitors error events.
Parameter and Description: Object res parameter, function listener, the listener function for error events.

Attribute	Type	Note
errMsg	string	Error Message

.offError

This method is used via UDPSocket.offError(function listener).

Feature Description: Removes the listener function for error events.
Parameter and Description: function listener, error event listener function, with the following arguments Object res:
Sample Code

```
const listener = function (res) { console.log(res) }
```

```
UDPSocket.onError(listener)
UDPSocket.offError(listener) // The same function object as the listener must be pa
```

.onListening

This method is used via UDPSocket.onListening(function listener).

Feature Description: Monitors events of starting to monitor data packet messages.
Parameter and Description: function listener, the listener function for events of starting to monitor data packet messages.

.offListening

This method is used via UDPSocket.offListening(function listener).

Feature Description: Removes the listener function for events of starting to monitor data packet messages.
Parameter and Description: function listener, the listener function passed in by onListening. If this parameter is not passed in, all listener functions will be removed.
Sample Code

```
const listener = function (res) { console.log(res) }

UDPSocket.onListening(listener)
UDPSocket.offListening(listener) // The same function object as the listener must b
```

.onMessage

This method is used via UDPSocket.onMessage(function listener).

Feature Description: Monitors events of receiving a message.
Parameter and Description: function listener, the listener function for the event that received the message, with the following arguments Object res:

Attribute	Type	Note
message	ArrayBuffer	The received message. The length of

		the message must be less than 4096.
remoteInfo	Object	Sender's address information

Structure attributes of "remoteInfo"

Structure attributes	Type	Note
address	string	The address of the socket sending the message
family	string	The protocol family in use, either IPv4 or IPv6.
Port No.	number	Port No.
size	number	The size of the message, measured in bytes.

.offMessage

This method is used via UDPSocket.offMessage(function listener).

Feature Description: Removes the listener function for events of receiving a message.

Parameter and Description: function listener, the listener function passed in by onMessage. If this parameter is not passed in, all listener functions will be removed.

Sample Code

```
const listener = function (res) { console.log(res) }

UDPSocket.onMessage(listener)
UDPSocket.offMessage(listener) // The same function object as the listener must be
```

.send

This method is used via UDPSocket.send(Object object).

Feature Description: Sends messages to the specified IP and port.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Note
address	string	-	Yes	The address to send the message to.
Port No.	number	-	Yes	Port number for message transmission
message	string/ArrayBuffer	-	Yes	Data to be transmitted
offset	number	0	No	The offset of the data to be sent, valid only when the message is of the ArrayBuffer type.
length	number	message.byteLength	No	The length of the data to be sent, valid only when the message is of the ArrayBuffer type.

Sample Code

```
const udp = wx.createUDPSocket()
  udp.bind()
  udp.send({
    address: '192.168.193.2',
    port: 8848,
    message: 'hello, how are you'
 })
```

.setTTL

This method is used via UDPSocket.setTTL(number ttl).

Feature Description: Sets the IP_TTL socket option, which determines the maximum number of hops allowed for an IP packet during transmission.

Parameter and Description: number ttl, ranging between 0 and 255.

Sample Code

```
const udp = wx.createUDPSocket()
udp.onListening(function () {
  udp.setTTL(64)
})
udp.bind()
```

.write

This method is used via UDPSocket.write().

Note:

Even if "connect" is called, it is necessary to input the address and port parameters in this interface.

Feature Description: Usage is the same as the send method, but not different from send if connect is not called beforehand.

Sample Code

```
const udp = wx.createUDPSocket()
udp.bind()
udp.connect({
  address: '192.168.193.2',
  port: 8848,
})
udp.write({
  address: '192.168.193.2',
  port: 8848,
  message: 'hello, how are you'
})
```

©2013-2025 Tencent Cloud International Pte. Ltd.

TCP Communication

Last updated：2025-03-20 18:01:56

createTCPSocket

This API is used via TCPSocket wx.createTCPSocket().

Feature description: Creates a TCP Socket instance.
Return Value: A TCP Socket instance.
Link limitations
Allows communication with non-local IP within the local area network.
Allows communication with configured server domain names.
Prohibits connections to the following port numbers: below 1024, 1099, 1433, 1521, 1719, 1720, 1723, 2049, 2375, 3128, 3306, 3389, 3659, 4045, 5060, 5061, 5432, 5984, 6379, 6000, 6566, 7001, 7002, 8000-8100, 8443, 8888, 9200, 9300, 10051, 10080, 11211, 27017, 27018, 27019.
A maximum of 20 TCPSockets can be created within every 5-minute interval.

TCPSocket

Note:
A TCP Socket instance, using IPv4 protocol by default.
When errCode is -2, the corresponding errno should be present in errMsg. Developers can examine the specific error message in the Linux code's errno-base.h and errno.h based on the errno.

.bindWifi

This method is used via TCPSocket.bindWifi(Object options).

Feature Description: Binds the TCP Socket to the current Wi-Fi network, which will, upon successful execution, trigger the onBindWifi event (supported only on Android).
Parameter and Description: Object options.

Attribute	Type	Default value	Required	Note

BSSID	string	-	Yes	The BSSID of the current Wi-Fi network, which can be obtained through wx.getConnectedWifi.

Sample Code

```
const tcp = wx.createTCPSocket()
tcp.bindWifi({ BSSID: 'xxx' })
tcp.onBindWifi(() => {})
```

.close

This method is used via TCPSocket.close().

Feature Description: Disables connection.

Sample Code

```
const tcp = wx.createTCPSocket()
tcp.close()
```

.connect

This method is used via TCPSocket.connect(Object options).

Feature Description: Initiates a connection on the specified socket.

Parameter and Description: Object options.

Attribute	Type	Default value	Required	Note
address	string	-	Yes	The address to which the socket is to be connected.
Port No.	number	-	Yes	The port to which the socket is to be connected.
timeout	number	2	No	The timeout duration for the socket connection, defaulting to 2 seconds.

Sample Code

```
const tcp = wx.createTCPSocket()
```

```
tcp.connect({address: '192.168.193.2', port: 8848})
```

.onClose

This method is used via TCPSocket.onClose(function listener).

Feature Description: Monitors events that are emitted once the socket is fully closed.
Parameter and Description: function listener, the listener function for events triggered once the socket is fully closed.

.offClose

This method is used via TCPSocket.offClose(function listener).

Feature Description: Removes the listener function for events that are emitted once the socket is fully closed.
Parameter and Description: function listener, the listener function passed in by onClose. If this parameter is not passed in, all listener functions will be removed.
Sample Code

```
const listener = function (res) { console.log(res) }

TCPSocket.onConnect(listener)
TCPSocket.offConnect(listener) // The same function object as the listener must be
```

.onConnect

This method is used via TCPSocket.onConnect(function listener).

Feature Description: Monitors events that are triggered when a socket connection is successfully established.
Parameter and Description: function listener, the listener function for events triggered when a socket connection is successfully established.

.offConnect

This method is used via TCPSocket.offConnect(function listener).

©2013-2025 Tencent Cloud International Pte. Ltd.

Feature Description: Removes the listener function for events triggered when a socket connection is successfully established.

Parameter and Description: function listener, the listener function passed in by onConnect. If this parameter is not passed in, all listener functions will be removed.

Sample Code

```
const listener = function (res) { console.log(res) }

TCPSocket.onConnect(listener)
TCPSocket.offConnect(listener) // The same function object as the listener must be
```

.onError

This method is used via TCPSocket.onError(function listener).

Feature Description: Monitoring is triggered when an error occurs.

Parameters and Description: The listener function of function listener.

Attribute	Type	Note
errMsg	string	Error Message

.offError

This method is used via TCPSocket.offError(function listener).

Feature Description: Removes the listener function that is triggered when an error occurs.

Parameter and Description: function listener, the listener function passed in by onError. If this parameter is not passed in, all listener functions will be removed.

Sample Code

```
const listener = function (res) { console.log(res) }

TCPSocket.onError(listener)
TCPSocket.offError(listener) // The same function object as the listener must be pa
```

.onMessage

This method is used via TCPSocket.onMessage(function listener).

Feature Description: Monitors events that are triggered when data is received.

Parameter and Description: function listener, when receiving data to trigger the event of the listener function, the parameter Object res as follows:

Attribute	Type	Note
message	ArrayBufffer	Received message
remoteInfo	Object	Sender's address information
localInfo	Object	Recipient's address information

Structure attributes of "remoteInfo"

Structure attributes	Type	Note
address	string	The address of the socket sending the message
family	string	The protocol family in use, either IPv4 or IPv6.
Port No.	number	Port No.
size	number	The size of the message, measured in bytes.

Structure attributes of "localInfo"

Structure attributes	Type	Note
address	string	The address of the socket sending the message
family	string	The protocol family in use, either IPv4 or IPv6.
Port No.	number	Port No.

.offMessage

This method is used via TCPSocket.offMessage(function listener).

Feature Description: Removes the listener function of the event that is triggered when data is received.

Parameter and Description: function listener, the listener function passed in by onMessage. If this parameter is not passed in, all listener functions will be removed.

Sample Code

```
const listener = function (res) { console.log(res) }

TCPSocket.onMessage(listener)
TCPSocket.offMessage(listener) // The same function object as the listener must be
```

.onBindWifi

This method is used via TCPSocket.onBindWifi(function listener).

Feature Description: Monitors events triggered when a socket successfully binds to the current Wi-Fi network.
Parameter and Description: function listener, the listener function for events triggered when a socket successfully binds to the current Wi-Fi network.

.offBindWifi

This method is used via TCPSocket.offBindWifi(function listener).

Feature Description: Removes the listener function for events triggered when a socket successfully binds to the current Wi-Fi network.
Parameters and Description: function listener, the listener function passed in by onBindWifi. If this parameter is not passed in, all listener functions will be removed.

Sample Code

```
const listener = function (res) { console.log(res) }

TCPSocket.onBindWifi(listener)
TCPSocket.offBindWifi(listener) // The same function object as the listener must be
```

.write

This method is used via TCPSocket.write(string|ArrayBuffer data).

Feature Description: Sends data over the socket.

Parameter and Description: string|ArrayBuffer data, the data to be sent.

Sample Code

```
const tcp = wx.createTCPSocket()
tcp.write('hello, how are you')
```

©2013-2025 Tencent Cloud International Pte. Ltd.

mDNS

Last updated：2025-03-25 18:15:55

stopLocalServiceDiscovery

The method of utilizing this API is wx.stopLocalServiceDiscovery(Object object).

Function Description: Cease the exploration of mDNS services.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.fail Callback Function Parameters: Object res.

Attributes	Types	Note
errMsg	string	Error message, with the valid value being 'task not found': invoked when stopLocalServiceDiscovery is called while not currently in the search service.

startLocalServiceDiscovery

The usage method for this API is wx.startLocalServiceDiscovery(Object object)

Note:

For iOS, the integration of the TMFMiniAppExtMDNS extension library is required to use the mDNS related interfaces.

wx.startLocalServiceDiscovery is a performance-consuming action. It will automatically stop after 30 seconds and execute the callback function registered with wx.onLocalServiceDiscoveryStop.
After invoking wx.startLocalServiceDiscovery, the next wx.startLocalServiceDiscovery can only be initiated once this search action has ceased. Actions to halt this search include invoking wx.stopLocalServiceDiscovery and the system automatically stopping the search after 30 seconds.
Function Description: Initiates the search for mDNS services within the local area network. The results of the search will be returned via the wx.onLocalService* event.
Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
serviceType	string	-	Supported	The type of service to be searched for
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.fail callback function parameters: Object res.

Attributes	Types	Note
errMsg	string	Error message, valid values are Invalid parameter: serviceType is empty. Scan task already exists: startLocalServiceDiscovery is called again while the search initiated by the current startLocalServiceDiscovery has not stopped.

Example Code

```
wx.startLocalServiceDiscovery({
    // There is a service of type _http._tcp. in the local area network connected t
    serviceType: '_http._tcp.',
    success: console.log,
    fail: console.log
})
```

offLocalServiceResolveFail

The usage method for this API is wx.offLocalServiceResolveFail(function listener).

Function Description: Removes the listener function for the event of mDNS service resolution failure.
Parameters and Description: function listener, the listening function passed in by onLocalServiceResolveFail. If this parameter is not passed, all listening functions will be removed.
Sample Code:

```
const listener = function (res) { console.log(res) }

wx.onLocalServiceResolveFail(listener)
wx.offLocalServiceResolveFail(listener) // The same function object as the listener
```

onLocalServiceResolveFail

The usage method for this API is wx.onLocalServiceResolveFail(function listener).

Function Description: Monitors the event of mDNS service resolution failure.
Parameters and Description: function listener, the listener function for the event that the mDNS service fails to resolve, with the following arguments Object res:

Attributes	Types	Note
serviceType	string	Service Type
serviceName	string	Service Name

offLocalServiceLost

The usage method for this API is wx.offLocalServiceLost(function listener).

Function Description: Removes the listening function for the event of mDNS service departure.

Parameters and Description: function listener, the listener passed in by onLocalServiceLost. If you don't pass this parameter, you can remove all the listener functions.

Sample Code:

```
const listener = function (res) { console.log(res) }

wx.onLocalServiceLost(listener)
wx.offLocalServiceLost(listener) // The same function objct as used during listenin
```

onLocalServiceLost

The usage method for this API is wx.onLocalServiceLost(function listener).

Function Description: Monitors the event of mDNS service departure.

Parameters and Description: function listener, the listener function for the event that the mDNS service leaves, with the following arguments Object res:

Attributes	Types	Note
serviceType	string	Service Type
serviceName	string	Service Name

offLocalServiceFound

The usage method for this API is wx.offLocalServiceFound(function listener).

Function Description: Removes the listening function for the event of mDNS service discovery.

Parameters and Description: function listener, onLocalServiceFound The listener function passed in. If this parameter is not passed, all listener functions are removed.

Sample Code:

```
const listener = function (res) { console.log(res) }

wx.onLocalServiceFound(listener)
wx.offLocalServiceFound(listener) // The same function object as the one used durin
```

©2013-2025 Tencent Cloud International Pte. Ltd.

onLocalServiceFound

The usage method for this API is wx.onLocalServiceFound(function listener).

Function Description: Monitors the event of mDNS service discovery.
Parameters and Description: function listener, the listener function for events discovered by the mDNS service, with the following arguments Object res:

Attributes	Types	Note
serviceType	string	Service Type
serviceName	string	Service Name
ip	string	The IP address of the service.
port	number	The port of the service.

offLocalServiceDiscoveryStop

The usage method for this API is wx.offLocalServiceDiscoveryStop(function listener).

Function Description: Removes the listening function for the event of mDNS service stop discovery.
Parameters and Description: function listener, onLocalServiceDiscoveryStop The listener function passed in. If this parameter is not passed, all listener functions are removed.
Sample Code:

```
const listener = function (res) { console.log(res) }

wx.onLocalServiceDiscoveryStop(listener)
wx.offLocalServiceDiscoveryStop(listener) // The same function object as the listen
```

onLocalServiceDiscoveryStop

The usage method for this API is wx.onLocalServiceDiscoveryStop(function listener).

Function Description: Monitors the event of mDNS service stop discovery.

Parameters and Description: function listener, the listener function for the event that the mDNS service stops searching.

Initiate request

Last updated：2025-02-24 17:10:35

request

This API is called using RequestTask wx.request(Object object).

Feature description: Initiates an HTTPS network request. Before using this API, please carefully read Network.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
url	string	-	True	The URL of the developer's server API
data	string/object/ArrayBuffer	-	False	Request parameter.
header	Object	-	False	Sets the request header. Referer cannot be set in the header. The default content-type is application/json
timeout	number	-	False	Timeout duration in milliseconds.
method	string	GET	False	HTTP request method.
dataType	string	json	False	The format of the returned data.
responseType	string	text	False	The type of the response data.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call

©2013-2025 Tencent Cloud International Pte. Ltd.

				ends (regardless of success or failure).

Valid values for object.method:

Value	Description
OPTIONS	HTTP OPTIONS request
GET	HTTP GET request
HEAD	HTTP HEAD request
POST	HTTP POST request
PUT	HTTP PUT request
DELETE	HTTP DELETE request
TRACE	HTTP TRACE request
CONNECT	HTTP CONNECT request

Valid values for object.dataType:

Value	Description
json	The returned data is in JSON format and will be parsed once using JSON.parse.
Others	The returned data will not be parsed using JSON.parse

Valid values for object.responseType

Value	Description
text	The response data is text.
arraybuffer	The response data is an ArrayBuffer.

object.success callback function

Parameter: Object res.

Property	Type	Description
data	string/Object/Arraybuffer	The data returned by the developer server.
statusCode	number	The HTTP status code returned by the developer server.

©2013-2025 Tencent Cloud International Pte. Ltd.

header	Object	The HTTP response header returned by the developer server.
cookies	Array.<string>	The cookies returned by the developer server in the format of string array (not supported in mini games).

object.fail callback function

Parameter: Object err

Property	Type	Description
errMsg	String	Error message.

Return value: RequestTask . Request task object.

Description of the data parameter:

The data ultimately sent to the server is of type String. If the passed data is not of type String, it will be converted to String. The conversion rules are as follows:

For `GET` method data, it will be converted to a query string (`encodeURIComponent(k)=encodeURIComponent(v)&encodeURIComponent(k)=encodeURIComponent(v)...)`

For `POST` method data with `header['content-type']` as `application/json` , it will be serialized into JSON.

For `POST` method data with `header['content-type']` as `application/x-www-form-urlencoded` , it will be converted into a query string `(encodeURIComponent(k)=encodeURIComponent(v)&encodeURIComponent(k)=encodeURIComponent(v)...)`

Example:

```
wx.request({
  url: 'test.php', // Example URL, not a real interface address
  data: {
    x: '',
    y: ''
  },
  header: {
    'content-type': 'application/json' // Default value
  },
  success(res) {
    console.log(res.data)
  }
})
```

RequestTask

.abort

This method is called using RequestTask.abort().

Feature description: Aborts a request task.

.onChunkReceived

This method is called using RequestTask.onChunkReceived(function listener).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Listens for the Transfer-Encoding Chunk Received event. It is triggered when a new chunk is received.

Parameter and description:function listener. The listener for the Transfer-Encoding Chunk Received event.

Parameter: Object res.

Property	Type	Description
res	Object	The response for each new chunk received from the developer's server.

res structure property

Structural property	Type	Description
data	ArrayBuffer	The returned chunk buffer.

.offChunkReceived

This method is called using RequestTask.offChunkReceived(function listener).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Removes the listener for the Transfer-Encoding Chunk Received event.

Parameter and description:function listener. The listener passed to onChunkReceived. If this parameter is not provided, all listeners will be removed.

Example

```
const listener = function (res) { console.log(res) }

RequestTask.onChunkReceived(listener)
RequestTask.offChunkReceived(listener) // Must pass the same function object used i
```

.onHeadersReceived

This method is called using RequestTask.onHeadersReceived(function listener).

Feature description: Listens for the HTTP Response Header event. The event is triggered before the response is received.

Parameter and description:function listener. The listener for the HTTP Response Header event.

Property	Type	Description
header	Object	The HTTP response header returned by the developer server.
statusCode	Number	The HTTP status code returned by the developer's server (currently not returned in the developer tools, but can be viewed on a real device)
cookies	Array.<string>	The cookies returned by the developer server in the format of string array.

.offHeadersReceived

This method is called using RequestTask.offHeadersReceived(function listener).

Feature description: Removes the listener for the HTTP Response Header event.

Parameter and description:function listener. The listener function passed to onHeadersReceived. If this parameter is not provided, all listeners will be removed.

Example:

```
const listener = function (res) { console.log(res) }
```

```
RequestTask.onHeadersReceived(listener)
RequestTask.offHeadersReceived(listener) // Must pass the same function object used
```

Upload

Last updated：2025-03-20 18:01:56

uploadFile

This API is used via UploadTask wx.uploadFile(Object object).

Feature Description: Uploads local resources to the server. The client initiates an HTTPS POST request, where the `content-type` is `multipart/form-data`. Prior to usage, read relevant instructions.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
url	string	-	Yes	Developer Server Address
filePath	string	-	Yes	Path to the file resource to be uploaded (local path)
name	string	-	Yes	The corresponding key for the file, through which developers can access the binary content of the file on the server side.
header	object	-	No	HTTP request's Header; setting Referer within the Header is not allowed.
formData	object	-	No	Additional form data in the HTTP request
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
data	string	Data returned from the developer's server.
statusCode	number	HTTP status code returned from the developer's server.

©2013-2025 Tencent Cloud International Pte. Ltd.

Return Value: UploadTask, an object that can monitor the progress of the upload and cancel the upload.

Sample Code

```
wx.chooseImage({
  success (res){
    const tempFilePaths = res.tempFilePaths
    wx.uploadFile({
      url: 'https://example.weixin.qq.com/upload', // This is merely an example and
      filePath: tempFilePaths[0],
      name: 'file',
      formData:{
        'user': 'test'
      },
      success (res){
        const data = res.data
        //do something
      }
    })
  }
})
```

UploadTask

.abort

This method is used via UploadTask.abort().

Feature Description: Aborts the upload task.

.onProgressUpdate

This method is used via UploadTask.onProgressUpdate(function listener).

Feature Description: Monitors upload progress change events.

Parameter and Description: function listener, listener for upload progress change events, arguments Object.res are as follows.

Attribute	Type	Description

progress	number	Download progress percentage
totalBytesWritten	number	Length of the data already downloaded, measured in Bytes.
totalBytesExpectedToWrite	number	Total length of the data expected to be downloaded, measured in bytes.

.offProgressUpdate

This method is used via UploadTask.offProgressUpdate(function listener).

Feature Description: Removes the listener function for upload progress change events.
Parameters and Description: function listener, the listener function passed in by onProgressUpdate. If this parameter is not passed in, all listener functions will be removed.
Sample Code

```
const listener = function (res) { console.log(res) }

DownloadTask.onProgressUpdate(listener)
DownloadTask.offProgressUpdate(listener) // The same function object as the listene
```

.onHeadersReceived

This method is used via UploadTask.onHeadersReceived(function listener).

Feature Description: Monitors HTTP Response Header events. This occurs earlier than the request completion event.
Parameter and Description: function listener, the listener function for HTTP Response Header events.

Attribute	Type	Description
header	Object	HTTP Response Header returned from the developer's server.

.offHeadersReceived

This method is used via UploadTask.offHeadersReceived(function listener).

©2013-2025 Tencent Cloud International Pte. Ltd.

Feature Description: Removes the listener function for HTTP Response Header events.

Parameters and Description: function listener, the listener function passed in by onHeadersReceived. If this parameter is not passed in, all listener functions will be removed.

Sample Code

```
const listener = function (res) { console.log(res) }

UploadTask.onHeadersReceived(listener)
UploadTask.offHeadersReceived(listener) // The same function object as the listener
```

Sample code

```
const uploadTask = wx.uploadFile({
  url: 'http://example.weixin.qq.com/upload', // This is merely an example and not
  filePath: tempFilePaths[0],
  name: 'file',
  formData:{
    'user': 'test'
  },
  success (res){
    const data = res.data
    //do something
  }
})

uploadTask.onProgressUpdate((res) => {
  console.log('Upload progress', res.progress)
  console.log('Length of already uploaded data', res.totalBytesSent)
  console.log('Total length of data expected to be uploaded', res.totalBytesExpecte
})

uploadTask.abort() // Aborts upload task
```

©2013-2025 Tencent Cloud International Pte. Ltd.

Websocket

Last updated：2025-03-20 18:01:56

Note:
It is advisable to utilize the SocketTask method to manage WebSocket connections, as it provides greater control over the lifecycle of each link. In scenarios where multiple WebSocket connections exist, the use of methods with a wx prefix may result in situations that deviate from expectations.

sendSocketMessage

The usage of this API is as follows: wx.sendSocketMessage(Object object)

Function Description: This feature allows data transmission via WebSocket connection. It requires an initial wx.connectSocket, and can only transmit data after the wx.onSocketOpen callback.
Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
data	string/ArrayBuffer	-	Supported	Content to be dispatched
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Sample Code:

```
let socketOpen = false
let socketMsgQueue = []
wx.connectSocket({
  url: 'test.php'
})

wx.onSocketOpen(function(res) {
```

```
  socketOpen = true
  for (let i = 0; i < socketMsgQueue.length; i++){
    sendSocketMessage(socketMsgQueue[i])
  }
  socketMsgQueue = []
})

function sendSocketMessage(msg) {
  if (socketOpen) {
    wx.sendSocketMessage({
      data:msg
    })
  } else {
    socketMsgQueue.push(msg)
  }
}
```

onSocketOpen

The usage method for this API is wx.onSocketOpen(function listener)

Function Description: Monitors the WebSocket connection opening event.
Parameters and Description: function listener, the monitoring function for the WebSocket connection opening event.

Attributes	Types	Note
header	object	HTTP response Header upon successful connection

onSocketMessage

The usage method for this API is wx.onSocketMessage(function listener)

Function Description: Monitors the event of WebSocket receiving a message from the server.
Parameters and Description: function listener, the monitoring function for the event of WebSocket receiving a message from the server.

Attributes	Types	Note

data	string/ArrayBuffer	Message returned by the server

onSocketError

The usage method for this API is wx.onSocketError(function listener)

Function Description: Monitors the WebSocket error event.

Parameters and Description: function listener, the monitoring function for the WebSocket error event.

Attributes	Types	Note
errMsg	string	Error Message

onSocketClose

The usage method for this API is wx.onSocketClose(function listener)

Function Description: Monitors the WebSocket connection closure event.

Parameters and Description: function listener, the monitoring function for the WebSocket connection closure event.

Attributes	Types	Note
code	number	A numerical value representing the status number of the closed connection, indicating the reason for the connection closure.
reason	string	A readable string indicating the reason for the connection closure.

connectSocket

The usage method for this API is SocketTask wx.connectSocket(Object object)

Function Description: Establishes a WebSocket connection. Please read the relevant instructions.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
url	string	-	Supported	Developer server's wss interface address
header	Object	-	Not required	HTTP Header, the Referer cannot be set in the Header.
protocols	Array.<string>	-	Not required	Array of sub-protocols.
tcpNoDelay	boolean	false	Not required	TCP_NODELAY setting during the establishment of a TCP connection.
timeout	number	-	Not required	Timeout duration, measured in milliseconds.
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Return Value: SocketTask.

Concurrency Count

In versions 1.5.0 and above, a maximum of five WebSocket connections can coexist simultaneously.

In versions below 1.5.0, a mini program can only maintain a single WebSocket connection at a time. If a WebSocket connection already exists, it will be automatically closed and a new WebSocket connection will be established.

Example Code

```
wx.connectSocket({
  url: 'wss://example.qq.com',
  header:{
    'content-type': 'application/json'
  },
  protocols: ['protocol1']
})
```

closeSocket

The usage of this API is wx.closeSocket(Object object).

Function Description: This function closes the WebSocket connection. The method of use is as follows.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
code	number	1000 (indicating a normal closure of the connection)	Not required	A numerical value representing the status number of the closed connection, indicating the reason for the connection closure.
reason	string	-	Not required	A readable string indicating the reason for the connection closure. This string must be a UTF-8 text that does not exceed 123 bytes in length (not characters).
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Sample Code:

```
wx.connectSocket({
  url: 'test.php'
})

//Please note that there may be timing issues here,
//If wx.connectSocket has not yet called back wx.onSocketOpen, and wx.closeSocket i
//It is imperative to call wx.closeSocket during the WebSocket's open period to eff
wx.onSocketOpen(function() {
  wx.closeSocket()
})

wx.onSocketClose(function(res) {
  console.log('WebSocket has been closed!')
})
```

SocketTask

.close

The method is utilized in the form of SocketTask.close(Object object).

Function Description: This function serves to close the WebSocket connection.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
code	number	1000 (indicating a normal closure of the connection)	Not required	A numerical value representing the status number of the closed connection, indicating the reason for the connection closure.
reason	string	-	Not required	A readable string indicating the reason for the connection closure. This string must be a UTF-8 text that does not exceed 123 bytes in length (not characters).
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.onClose

The method is employed as SocketTask.onClose(function listener).

Function Description: Monitors the WebSocket connection closure event.

Parameters and Description: function listener, the monitoring function for the WebSocket connection closure event.

Attributes	Types	Note
code	number	A numerical value representing the status number of the closed connection, indicating the reason for the connection closure.
reason	string	A readable string indicating the reason for the connection closure.

.onError

The method is invoked as SocketTask.onError(function listener).

Function Description: Monitors the WebSocket error event.
Parameters and Description: function listener, the monitoring function for the WebSocket error event.

Attributes	Types	Note
errMsg	string	Error Message

.onMessage

The method is implemented as SocketTask.onMessage(function listener).

Function Description: Monitors the event of WebSocket receiving a message from the server.
Parameters and Description: function listener, the monitoring function for the event of WebSocket receiving a message from the server.

Attributes	Types	Note
data	string/ArrayBuffer	Message returned by the server

.onOpen

The method is executed as SocketTask.onOpen(function listener).

Function Description: Monitors the WebSocket connection opening event.
Parameters and Description: function listener, the monitoring function for the WebSocket connection opening event.

Attributes	Types	Note
header	Object	HTTP response Header upon successful connection

.send

> The usage of this method is SocketTask.send(Object object).

Function Description: Data is transmitted via a WebSocket connection.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
data	string/ArrayBuffer	-	Supported	Content to be dispatched
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Download

Last updated：2025-03-20 18:01:56

downloadFile

This API is called using DownloadTask wx.downloadFile(Object object).

Note:

Please specify a reasonable Content-Type field in the server response header to ensure that the client processes the file type correctly.

Feature description: Downloads a file resource to the local device. The client initiates an HTTPS GET request, with the local temporary path (local path) of the file returned. The maximum file size allowed for a single download is 200 MB. Before calling this API, please read Network.

Parameter and description: Object object.

Property	Type	Required	Description
url	string	True	The URL of the resource to download.
header	Object	False	HTTP request header. The Referer cannot be set in the header.
filePath	string	False	The path where the downloaded file will be stored.
success	function	False	Callback function for successful API calls.
fail	function	False	Callback function for failed API calls.
complete	function	False	Callback function executed after API call ends (regardless of success or failure).

object.success callback function parameter: Object res.

Property	Type	Description
tempFilePath	string	Temporary file path (local path). Returned if filePath is not specified. The downloaded file will be stored in a temporary file
filePath	string	User file path (local path). Returned if filePath is specified and will match the provided filePath.
statusCode	number	The HTTP status code returned by the developer’s server.

Return value: DownloadTask

Example:

```
wx.downloadFile({
  url: 'https://example.com/audio/123', // Example URL, not a real resource
  success (res) {
    // As long as the server responds, the response content will be written to the
    if (res.statusCode === 200) {
      console.log('filePath', res.tempFilePath);
    }
  }
})
```

DownloadTask

.abort

This method is called using DownloadTask.abort().

Feature description: Aborts the download task.

.onProgressUpdate

This method is called using DownloadTask.onProgressUpdate(function listener).

Feature description: Listens for download progress change event.

Parameter and description:function listener. The listener for download progress change event.

Property	Type	Description
progress	number	Download progress percentage.
totalBytesWritten	number	The length of data that has been downloaded, in bytes.
totalBytesExpectedToWrite	number	The total length of data expected to be downloaded, in bytes.

.offProgressUpdate

This method is called using DownloadTask.offProgressUpdate(function listener).

Feature description: Removes the listener for download progress change event.
Parameter and description:function listener. The listener passed to onProgressUpdate. If not provided, all listeners will be removed.
Example:

```
const listener = function (res) { console.log(res) }

DownloadTask.onHeadersReceived(listener)
DownloadTask.offHeadersReceived(listener) // Must pass the same function object use
```

.onHeadersReceived

This method is called using DownloadTask.onHeadersReceived(function listener).

Feature description: Listens for the HTTP Response Header event. The event is triggered before the response is received.
Parameter and description:function listener. The listener for the HTTP Response Header event.

Property	Type	Description
header	Object	The HTTP response header returned by the developer's server.

.offHeadersReceived

This method is called using DownloadTask.offHeadersReceived(function listener).

Feature description: Removes the listener for the HTTP Response Header event.
Parameter and description:function listener. The listener passed to onHeadersReceived. If not provided, all listeners will be removed.
Example:

```
const listener = function (res) { console.log(res) }

DownloadTask.onHeadersReceived(listener)
DownloadTask.offHeadersReceived(listener) // Must pass the same function object use
```

©2013-2025 Tencent Cloud International Pte. Ltd.

Example

```
const downloadTask = wx.downloadFile({
  url: 'http://example.com/audio/123', // Example URL, not a real resource
  success (res) {
    wx.playVoice({
      filePath: res.tempFilePath
    })
  }
})

downloadTask.onProgressUpdate((res) => {
  console.log ('Download progress', res.progress)
  console.log (' Length of the downloaded data', res.totalBytesWritten)
  console.log ('Expected total length of the data to be downloaded', res.totalBytes
})

DownloadTask.abort () // Cancel the download task
```

Payment

Last updated：2025-06-03 18:00:36

requestPayment

This API is called using wx.requestPayment(Object object).

Feature description: This API initiates payment in the superapp. The superapp must have payment capabilities and complete the payment process integration. We recommend developers refer to our Payment and Merchant Practical Tutorial for API implementation. It requires the collaboration between superapp and mini program developers to effectively ensure data security and payment scalability.

Note: Generally, the parameters timeStamp, nonceStr, package, signType, and paySign should be generated by the mini program backend. The values are returned when the mini program frontend calls the backend order API, and then used to call wx.requestPayment to initiate the payment process.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
timeStamp	string	-	True	Timestamp in seconds since 1970-01-01 00:00:00, representing the current time.
nonceStr	string	-	True	Random string, up to 32 characters long.
package	string	-	True	The prepay_id parameter value returned from the unified order API, formatted as: prepay_id=***.
signType	string	-	True	Signature algorithm, should match the value used during backend ordering, typically: RSA.
paySign	string	-	True	Signature. For details, see Payment and Merchant Practical Tutorial
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

©2013-2025 Tencent Cloud International Pte. Ltd.

Example:

```
wx.requestPayment({
    timeStamp: '',
    nonceStr: '',
    package: '',
    signType: 'RSA',
    paySign: '',
    success (res) {
      /* res is typically returned by the superapp to the mini program frontend, wi
       * Upon successful payment, res.errMsg will be 'requestPayment:ok'
       * It is recommended for the mini program to call the backend API to confirm
      */
      if(res.errMsg === 'resquestPayment:ok'){
         // send request to miniprogram backend to comfirm payment state
      }
    },
    fail (err) {
      // The error message is usually returned by the superapp to the mini program
    }
})
```

If the superapp already has a mature and reliable payment process, the developers can customize the payment flow by combining this API with custom APIs.

©2013-2025 Tencent Cloud International Pte. Ltd.

Data Caching

Last updated：2025-02-24 17:10:36

setStorage

This API is called using wx.setStorage(Object object).

Feature description: Stores the data in local cache under the specified key, which will overwrite the original content associated with that key. The data is always available unless the user deletes it or the system cleans it up to free up storage space. The maximum size of data that can be stored in a single key is 1 MB, and the upper storage limit for all data is 10 MB.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
key	string	-	True	The key specified in the local cache.
data	any	-	True	The content that needs to be stored. Only objects of native types, dates, and objects that can be serialized via `JSON.stringify` are supported.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
wx.setStorage({
  key:"key",
  data:"value"
})
```

```
wx.setStorage({
  key: 'key',
  success (res) {
    console.log(res.data)
```

```
  }
})
```

setStorageSync

This API is called using wx.setStorageSync(string key, any data).

Note:
Storage should only be used for persistent storage of data, not for runtime data transfer or global state management. Excessive synchronous read/write operations during the startup process can significantly impact startup time.
Feature description: Stores the data in local cache under the specified key, which will overwrite the original content associated with that key. The data is always available unless the user deletes it or the system cleans it up to free up storage space. The maximum size of data that can be stored in a single key is 1 MB, and the upper storage limit for all data is 10 MB.
Parameters and description: string key, the key specified in the local cache;**any data**, content to be stored. Only objects of native types, dates, and objects that can be serialized via `JSON.stringify` are supported.
Example:

```
try {
  wx.setStorageSync('key', 'value')
} catch (e) { }
```

revokeBufferURL

Note:
This API is supported in mini programs but not in mini games.

This API is called using wx.revokeBufferURL(string url).

Feature description: Destroys the data in memory associated with the specified URL.
Parameter and description: string url. The URL of the binary data that needs to be destroyed.

removeStorage

This API is called using wx.removeStorage(Object object).

Feature description: Removes the specified key from the local cache.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
key	string	-	True	The key specified in the local cache.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
wx.removeStorage({
  key: 'key',
  success (res) {
    console.log(res)
  }
})
```

```
try {
  wx.removeStorageSync('key')
} catch (e) {
  // Do something when catch error
}
```

removeStorageSync

This API is called using wx.removeStorageSync(string key).

Feature description: Synchronously performs the same feature as wx.removeStorage..

Parameter and description:string key. The key specified in the local cache.

Example:

```
wx.removeStorage({
  key: 'key',
  success (res) {
    console.log(res)
  }
})
```

```
try {
  wx.removeStorageSync('key')
} catch (e) {
  // Do something when catch error
}
```

getStorage

This API is called using wx.getStorage(Object object).

Feature description: Asynchronously gets the content of the specified key from the local cache.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
key	string	-	True	The key specified in the local cache.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Object.success callback function parameter: Object res.

Property	Type	Description
data	any	The content corresponding to the key.

Example:

```
wx.getStorage({
```

```
  key: 'key',
  success (res) {
    console.log(res.data)
  }
})
```

```
wx.getStorage({
  key: 'key',
  success (res) {
    console.log(res.data)
  }
})
```

createBufferURL

Note:

This API is supported in mini programs but not in mini games.

This API is called using string wx.createBufferURL(ArrayBuffer|TypedArray buffer).

Feature description: Creates a unique URL in memory based on the provided buffer.

Parameter and description: ArrayBuffer|TypedArray buffer. The binary data to be stored in memory.

Return value:string.

getStorageSync

This API is called using any wx.getStorageSync(string key).

Note:

Storage should only be used for persistent storage of data, not for runtime data transfer or global state management.

Excessive synchronous read/write operations during the startup process can significantly impact startup time.

Feature description: Synchronously gets the content of the specified key from the local cache.

Parameter and description: string key. The key specified in the local cache.

Return value: any key. The content corresponding to the key.

Example:

```
try {
  var value = wx.getStorageSync('key')
  if (value) {
    // Do something with return value
  }
} catch (e) {
  // Do something when catch error
}
```

getStorageInfo

This API is called using wx.getStorageInfo(Object object).

Feature description: Asynchronously gets the information of the current storage.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.success callback function parameter: Object object.

Property	Type	Description
keys	Array.<string>	All keys in the current storage.
currentSize	number	Currently used space size in KB.
limitSize	number	Limited space size in KB.

Example:

```
wx.getStorageInfo({
  success(res) {
    console.log(res.keys)
    console.log(res.currentSize)
```

```
    console.log(res.limitSize)
  }
})
```

```
wx.getStorageInfo({
  success(res) {
    console.log(res.keys)
    console.log(res.currentSize)
    console.log(res.limitSize)
  }
})
```

getStorageInfoSync

This API is called using Object wx.getStorageInfoSync().

Feature description: Synchronously performs the same feature as wx.getStorageInfo.

Return value: Object object.

Property	Type	Description
keys	Array.	All keys in the current storage.
currentSize	number	Currently used space size in KB.
limitSize	number	Limited space size in KB.

Example:

```
wx.getStorageInfo({
  success (res) {
    console.log(res.keys)
    console.log(res.currentSize)
    console.log(res.limitSize)
  }
})
```

```
try {
  const res = wx.getStorageInfoSync()
  console.log(res.keys)
  console.log(res.currentSize)
```

```
  console.log(res.limitSize)
} catch (e) {
  // Do something when catch error
}
```

clearStorage

This API is called using wx.clearStorage(Object object).

Feature description: Clears the local data cache.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
wx.clearStorage()
```

```
try {
  wx.clearStorageSync()
} catch(e) {
  // Do something when catch error
}
```

clearStorageSync

This API is called using wx.clearStorageSync().

Feature description: Synchronously performs the same feature as wx.clearStorage.

Example:

```
wx.clearStorage()
```

```
try {
  wx.clearStorageSync()
} catch(e) {
  // Do something when catch error
}
```

batchSetStorage

This API is called using wx.batchSetStorage(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Stores the data in batches to the specified key in the local cache, which will overwrite the original content associated with that key. The data is always available unless the user deletes it or the system cleans it up to free up storage space. The maximum size of data that can be stored in a single key is 1 MB, and the upper storage limit for all data is 10 MB.

Parameter and description: Object object

Property	Type	Default value	Required	Description
kvList	Array	-	True	[{ key, value }]
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Array.<Object> kvList

Property	Type	Default value	Required	Description
key	string		True	The key specified in the local cache.
value	any		True	The content that needs to be stored. Only objects of native types, dates, and objects that can be serialized

				via JSON.stringify are supported.

Example:

```
wx.setStorage({
  key:"key",
  data:"value"
})
```

```
// Enable encrypted storage
wx.batchSetStorage({
  kvList: [{
    key: 'key',
    value: 'value',
  }],
})
```

batchSetStorageSync

This API is called using Array.<any> wx.batchGetStorageSync(Array.<string> keyList).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Stores the data in batches to the specified key in the local cache, which will overwrite the original content associated with that key. The data is always available unless the user deletes it or the system cleans it up to free up storage space. The maximum size of data that can be stored in a single key is 1 MB, and the upper storage limit for all data is 10 MB.

Parameter and description: Array.<Object> kvList。

Property	Type	Default value	Required	Description
key	string	-	True	The key specified in the local cache.
value	any	-	True	The content that needs to be stored. Only objects of native types, dates, and objects that can be serialized via JSON.stringify are supported.

Example:

```
try {
  wx.batchSetStorageSync([{key: 'key', value: 'value'}])
} catch (e) { }
```

batchGetStorage

This API is called using wx.batchGetStorage(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Asynchronously gets the content of the specified key in batches from the local cache.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
keyList	Array.<string>	-	True	The keyList specified in the local cache.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
wx.batchGetStorage({
  keyList: ['key'],
  success (res) {
    console.log(res)
  }
})
```

batchGetStorageSync

This API is called using Array.<any> wx.batchGetStorageSync(Array.<string> keyList).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Synchronously gets the content of the specified key in batches from the local cache.

Parameter and description: Array.<string> keyList. An array of specified keys in the local cache.

Return value: Array.<any>. The content corresponding to the key.

Example:

```
//Batch reading is more efficient than multiple getStorageSync calls when retrievin
try {
  var valueList = wx.batchGetStorageSync(['key'])
  if (valueList) {
    // Do something with return value
  }
} catch (e) {
  // Do something when catch error
}
```

Data Analysis

Last updated：2024-11-21 18:41:53

reportEvent

This API is used via wx.reportEvent(string eventId, object data).

Feature Description: Event Reporting.

Parameter and Description:

string eventId: The English name of the event set in the mp experimental system.

object data: An object that can be processed by JSON.stringify, which will be reported to the system concurrently.

©2013-2025 Tencent Cloud International Pte. Ltd.

Canvas

Canvas Overview

Last updated：2025-05-16 17:43:49

createOffscreenCanvas

This API is used via OffscreenCanvas wx.createOffscreenCanvas(object object, number width, number height, Object this).

Feature Description: Creates an instance of an offscreen canvas.

Parameter and Description:

number width: The width of the canvas.

number height: The height of the canvas.

object this: this of the current component instance under Custom Components

Object object

Attribute	Type	Default value	Required	Description
type	string	2d	No	Type of created offscreen canvas: 2d only.
width	number	-	No	Canvas Width
height	number	-	No	Canvas Height
compInst	Component	-	No	"This" of the current component instance in the context of custom components

createCanvasContext

This API is used via CanvasContext wx.createCanvasContext(string canvasId, Object this).

Feature Description: Creates a canvasContext object for canvas drawing context.

Parameter and Description:

string canvasId: The `canvas-id` attribute of the `<canvas>` component for which the context is to be obtained. Object this: In the context of custom components, "this" signifies the search for the `<canvas>` possessing the canvas-id within this custom component. If omitted, the search will not be conducted within any custom components.
Return Value: canvasContext.

canvasToTempFilePath

This API is used via wx.canvasToTempFilePath(Object object, Object this).

Feature Description: Exports the content of a specified area on the current canvas, generating an image of a specified size. Calling this method within the `draw()` callback ensures successful image export.
Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
x	number	0	No	Horizontal coordinate of the top left corner of the specified canvas area
y	number	0	No	Vertical coordinate of the top left corner of the specified canvas area
width	number	Canvas width - x	No	The width of the specified canvas area.
height	number	Canvas height - y	No	The height of the specified canvas area.
destWidth	number	Width multiplied by screen pixel density.	No	The width of the output image.
destHeight	number	Height multiplied by screen pixel density.	No	The height of the output image.
canvasId	string	-	Yes	Canvas identifier, canvas-id passed into the `<canvas>` component.
fileType	string	png	No	Type of the target file.

quality	number	-	Yes	The quality of the image, currently only effective for jpg. The value range is (0, 1]. If not within the range, it is treated as 1.0.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Valid values of object.fileType

Value	Description
jpg	JPG image
png	PNG image

Result of res callback for object.success

Attribute	Type	Description
tempFilePath	string	Temporary path to the generated file (local path)

Sample Code

```
wx.canvasToTempFilePath({
  x: 100,
  y: 200,
  width: 50,
  height: 50,
  destWidth: 100,
  destHeight: 100,
  canvasId: 'myCanvas',
  success(res) {
    console.log(res.tempFilePath)
  }
})
```

canvasPutImageData

This API is used via wx.canvasPutImageData(Object object, Object this).

Feature Description: Draws pixel data onto the canvas. In the context of a custom component, the second parameter passes in the custom component instance "this", to manipulate the `canvas` component in the component.

Parameter 1: Object.

Attribute	Type	Default value	Required	Description
canvasId	string	-	Yes	Canvas identifier, the attribute of the canvas-id passed into the `<canvas>` component.
data	Uint8ClampedArray	-	Yes	Image pixel data, a one-dimensional array, with every four items representing the RGBA of a pixel.
x	number	-	Yes	The position offset of the source image data in the target canvas (offset in the x-axis direction).
y	number	-	Yes	The position offset of the source image data in the target canvas (offset in the y-axis direction).
width	number	-	Yes	The width of the source image data's rectangular area.
height	number	-	Yes	The height of the source image data's rectangular area.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameter 2: Object this, under the custom component, the current component instance's this, to operate within the `<canvas>` component.

Sample Code

```
const data = new Uint8ClampedArray([255, 0, 0, 1])
wx.canvasPutImageData({
  canvasId: 'myCanvas',
  x: 0,
  y: 0,
  width: 1,
  data,
  success(res) {}
})
```

canvasGetImageData

This API is used via wx.canvasGetImageData(Object object, Object this).

Feature Description: Gets the implicit pixel data from the canvas area.

Parameter 1: Object.

Attribute	Type	Default value	Required	Description
canvasId	string	-	Yes	Canvas identifier, the attribute of the `canvas-id` passed into the component `<canvas>` .
x	number	-	Yes	The horizontal coordinate of the top-left corner of the rectangular area of the image data to be extracted.
y	number	-	Yes	The vertical coordinate of the top-left corner of the rectangular area of the image data to be extracted.
width	number	-	Yes	The width of the top-left corner of the rectangular area of the image data to be extracted.
height	number	-	Yes	The height of the top-left corner of the rectangular area of the image data to be extracted.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

©2013-2025 Tencent Cloud International Pte. Ltd.

Parameters for object.success callback function: Object res.

Attribute	Type	Description
width	number	The width of the image data rectangle.
height	number	The height of the source image data rectangle.
data	Uint8ClampedArray	Image pixel data, a one-dimensional array, with every four items representing the RGBA of a pixel.

Parameter 2：Object this, under Custom Components, this of the current component instance to manipulate <canvas> components within the component.

The parameter of the object.success callback function: Object this, "this" of the current component instance under custom components, used to operate the `<canvas>` component within.

Sample Code

```
wx.canvasGetImageData({
  canvasId: 'myCanvas',
  x: 0,
  y: 0,
  width: 100,
  height: 100,
  success(res) {
    console.log(res.width) // 100
    console.log(res.height) // 100
    console.log(res.data instanceof Uint8ClampedArray) // true
    console.log(res.data.length) // 100 * 100 * 4
  }
})
```

Image

Feature Description: Image object.

Parameter and Description

Attribute	Type	Description
src	string	The URL of the image
width	number	The actual width of the image
height	number	The actual height of the image

referrerPolicy	string	origin: Sends the complete referrer; no-referrer: Not to send. The format is fixed as https://servicewechat.com/{appid}/{version}/page-frame.html, where {appid} is the appid of the mini program, {version} is the version number of the mini program. A version number of 0 indicates a development version, trial version, or review version. A version number of devtools indicates a developer tool, and the rest are official versions.
onload	function	Callback function triggered upon the completion of image loading
onerror	function	Callback function triggered when an error occurs during image loading

ImageData

Feature Description: ImageData object.

Parameter and Description

Attribute	Type	Description
width	number	Describe the actual width of ImageData using pixels.
height	number	Describe the actual height of ImageData using pixels.
data	Uint8ClampedArray	A one-dimension array containing data in RGBA order, represented by integers ranging from 0 to 255 (inclusive).

canvasGradient

Feature Description: Gradient object.

.addColorStop

This method is used via CanvasGradient.addColorStop(number stop, Color color).

Feature Description: Adds a gradient point of color. Parts less than the minimum stop value will be rendered according to the color of the minimum stop value, and parts greater than the maximum stop value will be rendered according to the color of the maximum stop value, equivalent to CanvasGradient.addColorStop(number stop, Color color).

Parameter 1: number stop, indicating the position between the start and end of the gradient, range from 0 to 1.

Parameter 2: Color color, the color of the gradient point.

©2013-2025 Tencent Cloud International Pte. Ltd.

Sample Code

```
const ctx = wx.createCanvasContext('myCanvas')

// Create circular gradient
const grd = ctx.createLinearGradient(30, 10, 120, 10)
grd.addColorStop(0, 'red')
grd.addColorStop(0.16, 'orange')
grd.addColorStop(0.33, 'yellow')
grd.addColorStop(0.5, 'green')
grd.addColorStop(0.66, 'cyan')
grd.addColorStop(0.83, 'blue')
grd.addColorStop(1, 'purple')

// Fill with gradient
ctx.setFillStyle(grd)
ctx.fillRect(10, 10, 150, 80)
ctx.draw()
```

RenderingContext

Feature Description: Canvas drawing context.

The CanvasRenderingContext2D object can be obtained through the Canvas.getContext('2d') interface, implementing the properties and methods defined by the HTML Canvas 2D Context.

The WebGLRenderingContext object can be obtained through the Canvas.getContext('webgl') or OffscreenCanvas.getContext('webgl') interface, implementing all properties, methods, and constants defined by WebGL 1.0.

CanvasContext

For more details, see canvasContext.

OffscreenCanvas

For more details, see OffscreenCanvas.

Path2D

For more details, see Path2D.

Canvas

For more details, see Canvas.

Color

For more details, see Color.

©2013-2025 Tencent Cloud International Pte. Ltd.

OffscreenCanvas

Last updated：2025-05-16 17:44:18

Function Description: Offscreen canvas instances can be created through wx.createOffscreenCanvas.

Attribute

Attributes	Types	Note
width	number	Canvas Width
height	number	Canvas Height

Method Set

createImage

The method is utilized as Image OffscreenCanvas.createImage()

Function Description: This creates an image object. It is compatible with 2D Canvas.
Return Value: Image. Please note that it is advisable to create image objects using the canvas's own createImage.

getContext

The method is utilized as RenderingContext OffscreenCanvas.getContext(string contextType)

Function Description: This method returns the drawing context of the OffscreenCanvas.
Parameters and Description: string contextType, the type of drawing context, which needs to be consistent with the type passed in when creating OffscreenCanvas.

Value	Note
2d	2D type

©2013-2025 Tencent Cloud International Pte. Ltd.

Path2D

Last updated：2024-11-21 18:41:53

Function Description: The Path2D interface of the Canvas 2D API is utilized to declare paths, which will subsequently be employed by the CanvasRenderingContext2D object. The path methods of the CanvasRenderingContext2D interface are also present within the Path2D interface, permitting the creation of retainable and reusable paths in the Canvas as per necessity.

Attribute

Attributes	Types	Note
width	number	Describing the actual width of ImageData using pixels.
height	number	Describing the actual height of ImageData using pixels.
data	Uint8ClampedArray	A one-dimensional array containing data in RGBA order, represented by integers ranging from 0 to 255 (inclusive).

Method Set

addPath

The method is utilized as Path2D.addPath(Path2D path).

Function Description: Adds a path to the current path.
Parameters and Description: Path2D path, the Path2D path to be added.

arc

The method is employed as Path2D.arc(number x, number y, number radius, number startAngle, number endAngle, boolean counterclockwise).

Function Description: Incorporates an arc path segment.

Parameters and Descriptions:

number x: The x-coordinate of the circle's center.

number y: The y-coordinate of the circle's center.

number radius: The radius of the circle, which must be a positive number.

number startAngle: The starting angle of the arc.

number endAngle: The terminating angle of the arc.

boolean counterclockwise: Determines if the drawing is in a counterclockwise direction. If set to true, the drawing will commence from the endAngle and proceed to the startAngle.

arcTo

> The method is utilized as follows: Path2D.arcTo(number x1, number y1, number x2, number y2, number radius)

Function Description: Adds an arc path by specifying control points.

Parameters and Descriptions:

number x1: The x-coordinate of the first control point.

number y1: The y-coordinate of the first control point.

number x2: The x-coordinate of the second control point.

number y2: The y-coordinate of the second control point.

number radius: The radius of the circle, which must be a non-negative number.

bezierCurveTo

> The method is utilized as follows: Path2D.bezierCurveTo(number cp1x, number cp1y, number cp2x, number cp2y, number x, number y)

Function Description: Adds a cubic Bezier curve path.

Parameters and Descriptions:

number cp1x: The x-coordinate of the first control point.

number cp1y: The y-coordinate of the first control point.

number cp2x: The x-coordinate of the second control point.

number cp2y: The y-coordinate of the second control point.

number x: The x-coordinate of the endpoint.

number y: The y-coordinate of the endpoint.

©2013-2025 Tencent Cloud International Pte. Ltd.

closePath

The method is utilized as Path2D.closePath().

Function Description: Closes the path to the starting point.

ellipse

The method is utilized as Path2D.ellipse(number x, number y, number radiusX, number radiusY, number rotation, number startAngle, number endAngle, boolean counterclockwise).

Function Description: Adds an elliptical arc path.
Parameters and Descriptions:
number x: The x-coordinate of the ellipse's center.
number y: The y-coordinate of the ellipse's center.
number radiusX: The semi-major axis of the ellipse, must be a non-negative number.
number radiusY: The semi-minor axis of the ellipse, must be a non-negative number.
number rotation: The rotation angle of the ellipse.
number startAngle: The starting angle of the arc.
number endAngle: The terminating angle of the arc.
boolean counterclockwise: Determines if the drawing is in a counterclockwise direction. If set to true, the drawing will commence from the endAngle and proceed to the startAngle.

lineTo

The method is utilized as follows: Path2D.lineTo(number x, number y)

Function Description: Addition of a linear path.
Parameters and Descriptions:
number x: The x-coordinate of the endpoint.
number y: The y-coordinate of the endpoint.

moveTo

The method is utilized as follows: Path2D.moveTo(number x, number y)

Function Description: Movement of the path's starting point.

Parameters and Descriptions:

number x: The horizontal coordinate.

number y: The vertical coordinate.

quadraticCurveTo

The method is employed as follows: Path2D.quadraticCurveTo(number cpx, number cpy, number x, number y)

Function Description: Addition of a quadratic Bézier curve path.

Parameters and Descriptions:

number cpx: The horizontal coordinate of the control point.

number cpy: The vertical coordinate of the control point.

number x: The x-coordinate of the endpoint.

number y: The y-coordinate of the endpoint.

rect

The method is utilized as follows: Path2D.rect(number x, number y, number width, number height)

Function Description: Addition of a rectangular path.

Parameters and Descriptions:

number x: The horizontal coordinate of the starting point.

number y: The vertical coordinate of the starting point.

number width: The width of the rectangle, positive values extend to the right, negative values to the left.

number height: The height of the rectangle, positive values extend downwards, negative values upwards.

©2013-2025 Tencent Cloud International Pte. Ltd.

Canvas

Last updated：2025-03-25 18:15:55

Attributes

Canvas Width: The breadth of the canvas.
Canvas Height: The vertical extent of the canvas.

Method Set

cancelAnimationFrame

The method is utilized as follows: Canvas.cancelAnimationFrame(number requestID)

Function Description: Cancels the animation frame request scheduled by requestAnimationFrame. It is compatible with both 2D Canvas and WebGL Canvas, however, it does not support the concurrent use of 2D and WebGL methods.
Parameters and Description: Number requestID.

createImageData

The method is utilized as follows: Image Canvas.createImage()

Function Description: Creates an ImageData object. Only supported in 2D Canvas.
Parameters and Description: Image.

createImage

The method is utilized as follows: ImageData Canvas.createImageData()

Function Description: Creates an image object. Supports use under 2D Canvas and WebGL Canvas, but does not support mixing 2D and WebGL methods.

©2013-2025 Tencent Cloud International Pte. Ltd.

Parameters and Description: ImageData.

createPath2D

The method is utilized as follows: Path2D Canvas.createPath2D(Path2D path)

Function Description: Creates a Path2D object.
Parameters and Description: Path2D path.
Return Value: Path2D.

getContext

The method is utilized as follows: RenderingContext Canvas.getContext(string contextType)

Function Description: This method returns the drawing context of the Canvas.
Parameters and Description: string contextType, the context type.

Value	Note
2d	2D drawing context
webgl	WebGL drawing context
webgl2	WebGL2 drawing context

Return Value: RenderingContext, facilitating the acquisition of 2D and WebGL drawing contexts.

requestAnimationFrame

The method is utilized as follows: number Canvas.requestAnimationFrame(function callback)

Function Description: Executes upon the next redraw. Compatible with both 2D Canvas and WebGL Canvas, however, the methods of 2D and WebGL cannot be used interchangeably.
Parameters and Description: function callback, the callback to be executed.
Return Value: number, the request ID.

toDataURL

The method is employed as follows: string Canvas.toDataURL(string type, number encoderOptions)

Function Description: Returns a data URI that displays the image. The type parameter can be used to specify its format, with PNG as the default.

Parameters and Descriptions:

string type: The image format, default is image/png.

number encoderOptions: When the image format is specified as image/jpeg or image/webp, the image quality can be selected within the range of 0 to 1. If it exceeds the value range, the default value of 0.92 will be used. Other parameters will be disregarded.

Return Value: string.

canvasContext

Last updated：2025-03-25 18:15:56

Function Description: The drawing context for the canvas component. canvasContext is the old interface, the new Canvas 2D interface is Web-compliant.

Property Description

Attributes	Types	Note
fillStyle	String	Fill color. Usage is the same as CanvasContext.setFillStyle()
strokeStyle	String	Border color. The usage aligns with CanvasContext.setStrokeStyle().
font	String	-
globalCompositeOperation	String	The type of compositing operation applied when drawing a new shape. The current Android version only applies compositing to fill fill blocks, compositing effects used for stroke line segments are source-over. The currently supported operations are: Android：xor, source-over, source-atop, destination-out, lighter, overlay, darken, lighten, hard-light iOS：xor, source-over, source-atop, destination-over, destination-out, lighter, multiply, overlay, darken, lighten, color-dodge, color-burn, hard-light, soft-light, difference, exclusion, saturation, luminosity
shadowOffsetX	Number	The horizontal offset of the shadow relative to the shape.
shadowOffsetY	Number	The vertical offset of the shadow relative to the shape.
shadowColor	Number	The color of the shadow.
shadowBlur	Number	The blur level of the shadow.
lineWidth	Number	The width of the line. The usage is the same as CanvasContext.setLineWidth().
lineCap	Number	The style of the line's endpoints. The usage is the same as CanvasContext.setLineCap().

©2013-2025 Tencent Cloud International Pte. Ltd.

lineJoin	Number	The style of the line's intersection. The usage is the same as CanvasContext.setLineJoin().
miterLimit	Number	The maximum miter length. The usage is the same as CanvasContext.setMiterLimit().
lineDashOffset	Number	The offset of the dashed line, with an initial value of zero.
globalAlpha	Number	The global brush opacity. The range is 0-1, where 0 indicates complete transparency and 1 indicates complete opacity.

Legal values for lineJoin

Value	Description
bevel	bevel
round	round
miter	miter

Method Set

arc

The method is used as follows: CanvasContext.arc(number x, number y, number r, number sAngle, number eAngle, boolean counterclockwise).

Function Description: Creates an arc. The creation of a circle can specify the starting radian as 0 and the ending radian as 2 * Math.PI. Use the stroke or fill method to draw an arc in the canvas.
Parameters and Descriptions:
number x, the x-coordinate of the circle's center;
number y, the y-coordinate of the circle's center;
number r, the radius of the circle;
number sAngle, the starting radian, in radians (in the 3 o'clock direction);
number eAngle, the ending radian;
number counterclockwise, whether the direction of the radian is counterclockwise.

©2013-2025 Tencent Cloud International Pte. Ltd.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')

// Draw coordinates
ctx.arc(100, 75, 50, 0, 2 * Math.PI)
ctx.setFillStyle('#EEEEEE')
ctx.fill()

ctx.beginPath()
ctx.moveTo(40, 75)
ctx.lineTo(160, 75)
ctx.moveTo(100, 15)
ctx.lineTo(100, 135)
ctx.setStrokeStyle('#AAAAAA')
ctx.stroke()

ctx.setFontSize(12)
ctx.setFillStyle('black')
ctx.fillText('0', 165, 78)
ctx.fillText('0.5*PI', 83, 145)
ctx.fillText('1*PI', 15, 78)
ctx.fillText('1.5*PI', 83, 10)

// Draw points
ctx.beginPath()
ctx.arc(100, 75, 2, 0, 2 * Math.PI)
ctx.setFillStyle('lightgreen')
ctx.fill()

ctx.beginPath()
ctx.arc(100, 25, 2, 0, 2 * Math.PI)
ctx.setFillStyle('blue')
ctx.fill()

ctx.beginPath()
ctx.arc(150, 75, 2, 0, 2 * Math.PI)
ctx.setFillStyle('red')
ctx.fill()

// Draw arc
ctx.beginPath()
ctx.arc(100, 75, 50, 0, 1.5 * Math.PI)
ctx.setStrokeStyle('#333333')
ctx.stroke()

ctx.draw()
```

©2013-2025 Tencent Cloud International Pte. Ltd.

The three key coordinates for arc(100, 75, 50, 0, 1.5 * Math.PI) are as follows:
Green: Circle center (100, 75)
Red: Starting radian (0)
Blue: Ending radian (1.5 * Math.PI)

arcTo

The method is used as follows: CanvasContext.arcTo(number x1, number y1, number x2, number y2, number radius)

Function Description: Draws an arc path based on the control point and radius.
Parameters and Descriptions:
number x1, the x-axis coordinate of the first control point;
number y1, the y-axis coordinate of the first control point;
number x2, the x-axis coordinate of the second control point;
number y2, the y-axis coordinate of the second control point;
number radius, the radius of the arc.

draw

The method is used as follows: CanvasContext.draw(boolean reserve, function callback)

Function Description: Renders the previously described elements (paths, transformations, styles) in the drawing context onto the canvas.
Parameters and Descriptions:
boolean reserve: Determines whether the current drawing continues from the previous one. If the reserve parameter is set to false, the native layer will clear the canvas before proceeding with the drawing; if the reserve parameter is set to true, the current content on the canvas will be preserved, and the content drawn by the drawCanvas call will be overlaid on top, default is false.
function callback: The callback function executed upon completion of the drawing.
Sample Code: In the second draw() call, reserve is set to true. Therefore, the result of the previous drawing is preserved, and the fillStyle set in the context to 'red' has defaulted back to 'black'.

©2013-2025 Tencent Cloud International Pte. Ltd.

```
const ctx = wx.createCanvasContext('myCanvas')

ctx.setFillStyle('red')
ctx.fillRect(10, 10, 150, 100)
ctx.draw()
ctx.fillRect(50, 50, 150, 100)
ctx.draw(true)
```

drawImage

The method is utilized as follows: CanvasContext.drawImage(string imageResource, number sx, number sy, number sWidth, number sHeight, number dx, number dy, number dWidth, number dHeight)

Function Description: Draws an image onto the canvas.
Parameters and Descriptions:
string imageResource, the image resource to be drawn;
number sx, the x-coordinate of the top-left corner of the source image's rectangular selection box;
number sy, the y-coordinate of the top-left corner of the source image's rectangular selection box;
number sWidth, the width of the source image's rectangular selection box;
number sHeight, the height of the source image's rectangular selection box;
number dx, the x-axis position of the image's top-left corner on the target canvas;
number dy, the y-axis position of the image's top-left corner on the target canvas;
number dWidth, the width of the image drawn on the target canvas, allowing for scaling of the drawn image;
number dHeight, the height of the image drawn on the target canvas, allowing for scaling of the drawn image.
Sample Code: There are three versions of the syntax as follows.
drawImage(imageResource, dx, dy)
drawImage(imageResource, dx, dy, dWidth, dHeight)
drawImage(imageResource, sx, sy, sWidth, sHeight, dx, dy, dWidth, dHeight)

```
const ctx = wx.createCanvasContext('myCanvas')

wx.chooseImage({
  success(res) {
    ctx.drawImage(res.tempFilePaths[0], 0, 0, 150, 100)
    ctx.draw()
  }
```

```
})
```

createLinearGradient

The method is utilized as follows: CanvasGradient CanvasContext.createLinearGradient(number x0, number y0, number x1, number y1)

Function Description: Creates a linear gradient color. The returned CanvasGradient object requires the use of CanvasGradient.addColorStop() to specify gradient points, with at least two required.
Parameters and Descriptions:
number x0, the x-coordinate of the starting point;
number y0, the y-coordinate of the starting point;
number x1, the x-coordinate of the endpoint;
number y1, the y-coordinate of the endpoint.
Return Value: CanvasGradient.
Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')

// Create linear gradient
const grd = ctx.createLinearGradient(0, 0, 200, 0)
grd.addColorStop(0, 'red')
grd.addColorStop(1, 'white')

// Fill with gradient
ctx.setFillStyle(grd)
ctx.fillRect(10, 10, 150, 80)
ctx.draw()
```

createCircularGradient

The method is utilized as follows: CanvasGradient CanvasContext.createCircularGradient(number x, number y, number r)

Function Description: Creates a circular gradient color. The starting point is at the center of the circle, and the endpoint is on the circumference. The returned CanvasGradient object requires the use of

CanvasGradient.addColorStop() to specify gradient points, with at least two required.

Parameters and Descriptions:

number x, the x-coordinate of the circle's center;

number y, the y-coordinate of the circle's center;

number r, the radius of the circle.

Return Value: CanvasGradient.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')

// Create circular gradient
const grd = ctx.createCircularGradient(75, 50, 50)
grd.addColorStop(0, 'red')
grd.addColorStop(1, 'white')

// Fill with gradient
ctx.setFillStyle(grd)
ctx.fillRect(10, 10, 150, 80)
ctx.draw()
```

createPattern

Function Description: A method for creating a pattern for a specified image, which can repeat the original image in a specified direction. **The usage is as follows** CanvasContext.createPattern(string image, string repetition).

Parameters and Descriptions: string image, the source of the repeating image, supports code package paths and local temporary paths (local paths).

Value	Note
repeat	Repeats in both horizontal and vertical directions.
repeat-x	Repeats in the horizontal direction.
repeat-y	Repeats in the vertical direction.
no-repeat	Not repeating

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')
constpattern=ctx.createPattern('/path/to/image','repeat-x')
ctx.fillStyle=pattern
ctx.fillRect(0,0,300,150)
ctx.draw()
```

measureText

The usage of this method is Object CanvasContext.measureText(string text).

Function Description: Measures the dimensions of the text. Currently, it only returns the width of the text, a synchronous interface.

Parameters and Descriptions:

string image: The source of the repeating image, supporting both the code package path and the local temporary path (local path).

string repetition: How to repeat the image.

Legitimate values of repetition

Attributes	Types	Note
width	number	The width of the text.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')
ctx.font = 'italic bold 20px cursive'
const metrics = ctx.measureText('Hello World')
console.log(metrics.width)
```

save

The method is utilized as CanvasContext.save()

Function Description: Saves the drawing context.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')

// save the default fill style
ctx.save()
ctx.setFillStyle('red')
ctx.fillRect(10, 10, 150, 100)

// restore to the previous saved state
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
ctx.restore()
ctx.fillRect(50, 50, 150, 100)

ctx.draw()
```

restore

The method is utilized as CanvasContext.restore()

Function Description: Restore the previously saved drawing context.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')

// save the default fill style
ctx.save()
ctx.setFillStyle('red')
ctx.fillRect(10, 10, 150, 100)

// restore to the previous saved state
ctx.restore()
ctx.fillRect(50, 50, 150, 100)

ctx.draw()
```

beginPath

The method is utilized as CanvasContext.beginPath()

Function Description: Initiate the creation of a path. It necessitates the invocation of fill or stroke to utilize the path for filling or outlining.

At the outset, it is equivalent to invoking beginPath once.

Within the same path, multiple settings such as setFillStyle, setStrokeStyle, setLineWidth, etc., the final setting prevails.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')
// begin path
```

```
ctx.rect(10, 10, 100, 30)
ctx.setFillStyle('yellow')
ctx.fill()

// begin another path
ctx.beginPath()
ctx.rect(10, 40, 100, 30)

// only fill this rect, not in current path
ctx.setFillStyle('blue')
ctx.fillRect(10, 70, 100, 30)

ctx.rect(10, 100, 100, 30)

// it will fill current path
ctx.setFillStyle('red')
ctx.fill()
ctx.draw()
```

closePath

The method is utilized as CanvasContext.closePath()

Function Description: Close a path. This will connect the start and end points. If the path is closed and fill or stroke is not invoked, and a new path is initiated, the previous path will not be rendered.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')
ctx.moveTo(10, 10)
ctx.lineTo(100, 10)
ctx.lineTo(100, 100)
ctx.closePath()
ctx.stroke()
ctx.draw()
```

```
const ctx = wx.createCanvasContext('myCanvas')
// begin path
ctx.rect(10, 10, 100, 30)
ctx.closePath()

// begin another path
```

```
ctx.beginPath()
ctx.rect(10, 40, 100, 30)

// only fill this rect, not in current path
ctx.setFillStyle('blue')
ctx.fillRect(10, 70, 100, 30)

ctx.rect(10, 100, 100, 30)

// it will fill current path
ctx.setFillStyle('red')
ctx.fill()
ctx.draw()
```

moveTo

The method is utilized as CanvasContext.moveTo(number x, number y)

Function Description: Move the path to a specified point in the canvas, without the need to create a line. Utilize the stroke method to draw the line.

Parameters and Description:

number x, the x-coordinate of the target position;

number y, the y-coordinate of the target position.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')
ctx.moveTo(10, 10)
ctx.lineTo(100, 10)

ctx.moveTo(10, 50)
ctx.lineTo(100, 50)
ctx.stroke()
ctx.draw()
```

lineTo

The method is utilized as CanvasContext.lineTo(number x, number y)

Function Description: Add a new point, then create a line from the last specified point to the target point, using the stroke method to draw the line.

Parameters and Description:

number x, the x-coordinate of the target position;

number y, the y-coordinate of the target position.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')
ctx.moveTo(10, 10)
ctx.rect(10, 10, 100, 50)
ctx.lineTo(110, 60)
ctx.stroke()
ctx.draw()
```

quadraticCurveTo

The method is utilized as CanvasContext.quadraticCurveTo(number cpx, number cpy, number x, number y)

Function Description: Create a quadratic Bézier curve path. The starting point of the curve is the previous point in the path.

Parameters and Description:

number cpx, the x-coordinate of the Bézier control point;

number cpy, the y-coordinate of the Bézier control point;

number x, the x-coordinate of the end point;

number y, the y-coordinate of the end point.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')

// Draw points
ctx.beginPath()
ctx.arc(20, 20, 2, 0, 2 * Math.PI)
ctx.setFillStyle('red')
ctx.fill()

ctx.beginPath()
ctx.arc(200, 20, 2, 0, 2 * Math.PI)
ctx.setFillStyle('lightgreen')
ctx.fill()
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
ctx.beginPath()
ctx.arc(20, 100, 2, 0, 2 * Math.PI)
ctx.setFillStyle('blue')
ctx.fill()

ctx.setFillStyle('black')
ctx.setFontSize(12)

// Draw guides
ctx.beginPath()
ctx.moveTo(20, 20)
ctx.lineTo(20, 100)
ctx.lineTo(200, 20)
ctx.setStrokeStyle('#AAAAAA')
ctx.stroke()

// Draw quadratic curve
ctx.beginPath()
ctx.moveTo(20, 20)
ctx.quadraticCurveTo(20, 100, 200, 20)
ctx.setStrokeStyle('black')
ctx.stroke()

ctx.draw()
```

The three key coordinates for moveTo(20, 20) quadraticCurveTo(20, 100, 200, 20) are as follows:

Red: Starting point (20, 20)

Blue: Control point (20, 100)

Green: Termination point (200, 20)

bezierCurveTo

> The method is used as follows: CanvasContext.bezierCurveTo(number cp1x, number cp1y, number cp2x, number cp2y, number x, number y)

Function Description: Creates a cubic Bezier curve path. The starting point of the curve is the previous point in the path.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')

// Draw points
```

```
ctx.beginPath()
ctx.arc(20, 20, 2, 0, 2 * Math.PI)
ctx.setFillStyle('red')
ctx.fill()

ctx.beginPath()
ctx.arc(200, 20, 2, 0, 2 * Math.PI)
ctx.setFillStyle('lightgreen')
ctx.fill()

ctx.beginPath()
ctx.arc(20, 100, 2, 0, 2 * Math.PI)
ctx.arc(200, 100, 2, 0, 2 * Math.PI)
ctx.setFillStyle('blue')
ctx.fill()

ctx.setFillStyle('black')
ctx.setFontSize(12)

// Draw guides
ctx.beginPath()
ctx.moveTo(20, 20)
ctx.lineTo(20, 100)
ctx.lineTo(150, 75)

ctx.moveTo(200, 20)
ctx.lineTo(200, 100)
ctx.lineTo(70, 75)
ctx.setStrokeStyle('#AAAAAA')
ctx.stroke()

// Draw quadratic curve
ctx.beginPath()
ctx.moveTo(20, 20)
ctx.bezierCurveTo(20, 100, 200, 100, 200, 20)
ctx.setStrokeStyle('black')
ctx.stroke()

ctx.draw()
```

The three key coordinates for moveTo(20, 20) bezierCurveTo(20, 100, 200, 100, 200, 20) are as follows:

Red: Starting point (20, 20)

Blue: Two control points (20, 100) (200, 100)

Green: Termination point (200, 20)

©2013-2025 Tencent Cloud International Pte. Ltd.

rect

The method is used as follows: CanvasContext.rect(number x, number y, number width, number height)

Function Description: Creates a rectangular path. The rectangle must be actually drawn on the canvas using the fill or stroke method.

Parameters and Descriptions:

number x, the x-coordinate of the top-left corner of the rectangular path;

number y, the y-coordinate of the top-left corner of the rectangular path;

number width, the width of the rectangular path;

number height, the height of the rectangular path.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')
ctx.rect(10, 10, 150, 75)
ctx.setFillStyle('red')
ctx.fill()
ctx.draw()
```

clip

The method is used as follows: CanvasContext.clip()

Function Description: Cuts any shape and size from the original canvas. Once an area is cut, all subsequent drawings will be confined within the cut area (unable to access other areas on the canvas). The current canvas area can be saved before using the clip method by using the save method, and can be restored at any time in the future using the restore method.

Example Code

```
const ctx = wx.createCanvasContext('myCanvas')

wx.downloadFile({
  url: 'https://is5.mzstatic.com/image/thumb/Purple128/v4/75/3b/90/753b907c-b7fb-58
  success(res) {
    ctx.save()
    ctx.beginPath()
    ctx.arc(50, 50, 25, 0, 2 * Math.PI)
    ctx.clip()
```

```
    ctx.drawImage(res.tempFilePath, 25, 25)
    ctx.restore()
    ctx.draw()
  }
})
```

stroke

The method is used as follows: CanvasContext.stroke()

Function Description: Draws the border of the current path. The default color is black.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')
ctx.moveTo(10, 10)
ctx.lineTo(100, 10)
ctx.lineTo(100, 100)
ctx.stroke()
ctx.draw()
```

The path depicted by stroke() is calculated from beginPath(), but it does not include strokeRect().

```
const ctx = wx.createCanvasContext('myCanvas')
// begin path
ctx.rect(10, 10, 100, 30)
ctx.setStrokeStyle('yellow')
ctx.stroke()

// begin another path
ctx.beginPath()
ctx.rect(10, 40, 100, 30)

// only stoke this rect, not in current path
ctx.setStrokeStyle('blue')
ctx.strokeRect(10, 70, 100, 30)

ctx.rect(10, 100, 100, 30)

// it will stroke current path
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
ctx.setStrokeStyle('red')
ctx.stroke()
ctx.draw()
```

strokeRect

The method is used as follows: CanvasContext.strokeRect(number x, number y, number width, number height)

Function Description: Draws a rectangle (non-filled). The color of the rectangle's outline is set using setStrokeStyle, if not set, the default color is black.

Parameters and Descriptions:

number x, the x-coordinate of the top-left corner of the rectangular path;

number y, the y-coordinate of the top-left corner of the rectangular path;

number width, the width of the rectangular path;

number height, the height of the rectangular path.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')
ctx.setStrokeStyle('red')
ctx.strokeRect(10, 10, 150, 75)
ctx.draw()
```

strokeText

The method is used as follows: CanvasContext.strokeText(string text, number x, number y, number maxWidth)

Function Description: A method to draw text outlines at the given (x, y) position.

Parameters and Description:

string text, the text to be drawn;

number x, the x-axis coordinate of the text's starting point;

number y, the y-axis coordinate of the text's starting point;
number maxWidth, the maximum width to be drawn, optional.

clearRect

The method is used as follows: CanvasContext.clearRect(number x, number y, number width, number height)

Function Description: Clears the content within the specified rectangular area on the canvas.
Parameters and Descriptions:
number x, the x-coordinate of the top-left corner of the rectangular path;
number y, the y-coordinate of the top-left corner of the rectangular path;
number width, the width of the rectangular path;
number height, the height of the rectangular path.
Example Code: clearRect does not draw a white rectangle in the specified area, but rather clears it. For a more intuitive understanding, a background color has been added to the canvas.

```
<canvas canvas-id="myCanvas" style="border: 1px solid; background: #123456;" />
```

```
const ctx = wx.createCanvasContext('myCanvas')
ctx.setFillStyle('red')
ctx.fillRect(0, 0, 150, 200)
ctx.setFillStyle('blue')
ctx.fillRect(150, 0, 150, 200)
ctx.clearRect(10, 10, 150, 75)
ctx.draw()
```

fill

The method is used as follows: CanvasContext.fill()

Function Description: Fills the content within the current path. The default fill color is black.
Sample Code: If the current path is not closed, the fill() method will connect the start and end points, and then fill it.

```
const ctx = wx.createCanvasContext('myCanvas')
```

```
ctx.moveTo(10, 10)
ctx.lineTo(100, 10)
ctx.lineTo(100, 100)
ctx.fill()
ctx.draw()
```

The path filled by fill() is calculated from beginPath(), but it does not include fillRect().

```
const ctx = wx.createCanvasContext('myCanvas')
// begin path
ctx.rect(10, 10, 100, 30)
ctx.setFillStyle('yellow')
ctx.fill()

// begin another path
ctx.beginPath()
ctx.rect(10, 40, 100, 30)

// only fill this rect, not in current path
ctx.setFillStyle('blue')
ctx.fillRect(10, 70, 100, 30)

ctx.rect(10, 100, 100, 30)

// it will fill current path
ctx.setFillStyle('red')
ctx.fill()
ctx.draw()
```

fillRect

The method is used as follows: CanvasContext.fillRect(number x, number y, number width, number height)

Function Description: Fills a rectangle. The fill color of the rectangle is set using setFillStyle, if not set, the default is black.

Parameters and Descriptions:

number x, the x-coordinate of the top-left corner of the rectangular path;
number y, the y-coordinate of the top-left corner of the rectangular path;
number width, the width of the rectangular path;
number height, the height of the rectangular path.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')
ctx.setFillStyle('red')
ctx.fillRect(10, 10, 150, 75)
ctx.draw()
```

fillText

The method is used as follows: CanvasContext.fillText(string text, number x, number y, number maxWidth)

Function Description: Draws filled text on the canvas.

Parameters and Description:

string text, the text output on the canvas;
number x, the x-coordinate position of the top left corner of the drawn text;
number y, the y-coordinate position of the top left corner of the drawn text;
number maxWidth, the maximum width that needs to be drawn, optional.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')
```

```
const ctx = wx.createCanvasContext('myCanvas')

ctx.setFontSize(20)
ctx.fillText('Hello', 20, 20)
ctx.fillText('MINA', 100, 100)

ctx.draw()
```

©2013-2025 Tencent Cloud International Pte. Ltd.

scale

The method is used as follows: CanvasContext.scale(number scaleWidth, number scaleHeight)

Function Description: After calling, the x and y coordinates of the path created thereafter will be scaled. Multiple calls will multiply the factors.

Parameters and Description:

number scaleWidth, the scaling factor for the x-coordinate (1 = 100%, 0.5 = 50%, 2 = 200%);

number scaleHeight, the scaling factor for the y-coordinate (1 = 100%, 0.5 = 50%, 2 = 200%).

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')

ctx.strokeRect(10, 10, 25, 15)
ctx.scale(2, 2)
ctx.strokeRect(10, 10, 25, 15)
ctx.scale(2, 2)
ctx.strokeRect(10, 10, 25, 15)

ctx.draw()
```

rotate

The method is used as follows: CanvasContext.rotate(number rotate)

Function Description: Rotates the current coordinate axis clockwise with the origin as the center. Multiple calls will accumulate the rotation angle. The origin can be modified using the translate method.

Parameters and Description: number rotate, rotation angle, calculated in radians as degrees * Math.PI/180; degrees range from 0-360.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')

ctx.strokeRect(100, 10, 150, 100)
ctx.rotate(20 * Math.PI / 180)
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
ctx.strokeRect(100, 10, 150, 100)
ctx.rotate(20 * Math.PI / 180)
ctx.strokeRect(100, 10, 150, 100)

ctx.draw()
```

transform

The method is used as follows: CanvasContext.transform(number scaleX, number skewX, number skewY, number scaleY, number translateX, number translateY)

Function Description: A method to repeatedly overlay the current transformation using a matrix.
Parameters and Description:
number scaleX, horizontal scaling;
number scaleY, vertical scaling;
number skewX, horizontal skewing;
number skewY, vertical skewing;
number translateX, horizontal translation;
number translateY, vertical translation.

translate

The method is used as follows: CanvasContext.translate(number x, number y)

Function Description: Transforms the origin (0, 0) of the current coordinate system. The default coordinate system origin is the top left corner of the page.
Parameters and Description:
number x, horizontal coordinate translation;
number y, vertical coordinate translation.
Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')

ctx.strokeRect(10, 10, 150, 100)
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
ctx.translate(20, 20)
ctx.strokeRect(10, 10, 150, 100)
ctx.translate(20, 20)
ctx.strokeRect(10, 10, 150, 100)

ctx.draw()
```

setTransform

The method is used as follows: CanvasContext.setTransform(number scaleX, number skewX, number skewY, number scaleY, number translateX, number translateY)

Function Description: A method to reset (overwrite) the current transformation using a matrix.
Parameters and Description:
number scaleX, horizontal scaling;
number scaleY, vertical scaling;
number skewX, horizontal skewing;
number skewY, vertical skewing;
number translateX, horizontal translation;
number translateY, vertical translation.

setFillStyle

The method is used as follows: CanvasContext.setFillStyle(string|CanvasGradient color)

Function Description: Set the fill color.
Parameters and Description: string CanvasGradient color, the fill color, with the default color being black.
Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')
ctx.setFillStyle('red')
ctx.fillRect(10, 10, 150, 75)
ctx.draw()
```

©2013-2025 Tencent Cloud International Pte. Ltd.

setStrokeStyle

The method is used as follows: CanvasContext.setStrokeStyle(string|CanvasGradient color)

Function Description: Set the stroke color.

Parameters and Description: string CanvasGradient color, the fill color, with the default color being black.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')
ctx.setStrokeStyle('red')
ctx.strokeRect(10, 10, 150, 75)
ctx.draw()
```

setShadow

The method is used as follows: CanvasContext.setShadow(number offsetX, number offsetY, number blur, string color)

Function Description: Set the shadow style.

Parameters and Description:

number offsetX, the horizontal offset of the shadow relative to the shape, default value is 0;

number offsetY, the vertical offset of the shadow relative to the shape, default value is 0;

number blur, the blur level of the shadow, the larger the value, the more blurred it is. Range 0-100, default value is 0;

string color, the color of the shadow. Default value is black.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')
ctx.setFillStyle('red')
ctx.setShadow(10, 50, 50, 'blue')
ctx.fillRect(10, 10, 150, 75)
ctx.draw()
```

setGlobalAlpha

The method is used as follows: CanvasContext.setGlobalAlpha(number alpha)

Function Description: Set the global brush transparency.

Parameters and Description: number alpha, transparency. Range 0-1, 0 indicates complete transparency, 1 indicates complete opacity.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')

ctx.setFillStyle('red')
ctx.fillRect(10, 10, 150, 100)
ctx.setGlobalAlpha(0.2)
ctx.setFillStyle('blue')
ctx.fillRect(50, 50, 150, 100)
ctx.setFillStyle('yellow')
ctx.fillRect(100, 100, 150, 100)

ctx.draw()
```

setLineWidth

The method is used as follows: CanvasContext.setLineWidth(number lineWidth)

Function Description: Set the width of the line.

Parameters and Description: number lineWidth, the width of the line, unit: px.

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')
ctx.beginPath()
ctx.moveTo(10, 10)
ctx.lineTo(150, 10)
ctx.stroke()
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
ctx.beginPath()
ctx.setLineWidth(5)
ctx.moveTo(10, 30)
ctx.lineTo(150, 30)
ctx.stroke()

ctx.beginPath()
ctx.setLineWidth(10)
ctx.moveTo(10, 50)
ctx.lineTo(150, 50)
ctx.stroke()

ctx.beginPath()
ctx.setLineWidth(15)
ctx.moveTo(10, 70)
ctx.lineTo(150, 70)
ctx.stroke()

ctx.draw()
```

setLineJoin

The method is used as follows: CanvasContext.setLineJoin(string lineJoin)

Function Description: Set the style of the line intersection.

Parameters and Description: string lineJoin, the style of the line's end intersection.

Value	Note
bevel	Bevel
round	Rounded corner
miter	Sharp corner

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')
ctx.beginPath()
```

```
ctx.moveTo(10, 10)
ctx.lineTo(100, 50)
ctx.lineTo(10, 90)
ctx.stroke()

ctx.beginPath()
ctx.setLineJoin('bevel')
ctx.setLineWidth(10)
ctx.moveTo(50, 10)
ctx.lineTo(140, 50)
ctx.lineTo(50, 90)
ctx.stroke()

ctx.beginPath()
ctx.setLineJoin('round')
ctx.setLineWidth(10)
ctx.moveTo(90, 10)
ctx.lineTo(180, 50)
ctx.lineTo(90, 90)
ctx.stroke()

ctx.beginPath()
ctx.setLineJoin('miter')
ctx.setLineWidth(10)
ctx.moveTo(130, 10)
ctx.lineTo(220, 50)
ctx.lineTo(130, 90)
ctx.stroke()

ctx.draw()
```

setLineCap

The method is used as follows: CanvasContext.setLineCap(string lineCap)

Function Description: Sets the style of the line's endpoint.

Parameters and Description: string lineGap, the style of the line's termination junction.

Value	Note

butt	Adding flat edges to each end of the line
round	Adding round caps to each end of the line
square	Adding square caps to each end of the line

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')
ctx.beginPath()
ctx.moveTo(10, 10)
ctx.lineTo(150, 10)
ctx.stroke()

ctx.beginPath()
ctx.setLineCap('butt')
ctx.setLineWidth(10)
ctx.moveTo(10, 30)
ctx.lineTo(150, 30)
ctx.stroke()

ctx.beginPath()
ctx.setLineCap('round')
ctx.setLineWidth(10)
ctx.moveTo(10, 50)
ctx.lineTo(150, 50)
ctx.stroke()

ctx.beginPath()
ctx.setLineCap('square')
ctx.setLineWidth(10)
ctx.moveTo(10, 70)
ctx.lineTo(150, 70)
ctx.stroke()

ctx.draw()
```

setLineDash

The method is used as follows: CanvasContext.setLineDash(Array.<number> pattern, number offset)

Function Description: Set the style of the dashed line.

Parameters and Description: Array.<number> pattern, a set of numbers describing the alternating lengths of the drawn segments and gaps (in coordinate space units); number offset, the offset of the dashed line.
Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')

ctx.setLineDash([10, 20], 5);

ctx.beginPath();
ctx.moveTo(0,100);
ctx.lineTo(400, 100);
ctx.stroke();

ctx.draw()
```

setMiterLimit

The method is used as follows: CanvasContext.setMiterLimit(number miterLimit)

Function Description: Set the maximum miter length. The miter length refers to the distance between the inner and outer angles at the intersection of two lines. It is only effective when CanvasContext.setLineJoin() is set to miter. If it exceeds the maximum miter length, the junction will be displayed as bevel in lineJoin.
Parameters and Description: number miterLimit, the maximum miter length.
Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')
ctx.beginPath()
ctx.setLineWidth(10)
ctx.setLineJoin('miter')
ctx.setMiterLimit(1)
ctx.moveTo(10, 10)
ctx.lineTo(100, 50)
ctx.lineTo(10, 90)
ctx.stroke()

ctx.beginPath()
ctx.setLineWidth(10)
ctx.setLineJoin('miter')
ctx.setMiterLimit(2)
ctx.moveTo(50, 10)
ctx.lineTo(140, 50)
ctx.lineTo(50, 90)
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
ctx.stroke()

ctx.beginPath()
ctx.setLineWidth(10)
ctx.setLineJoin('miter')
ctx.setMiterLimit(3)
ctx.moveTo(90, 10)
ctx.lineTo(180, 50)
ctx.lineTo(90, 90)
ctx.stroke()

ctx.beginPath()
ctx.setLineWidth(10)
ctx.setLineJoin('miter')
ctx.setMiterLimit(4)
ctx.moveTo(130, 10)
ctx.lineTo(220, 50)
ctx.lineTo(130, 90)
ctx.stroke()

ctx.draw()
```

setFontSize

The method is used as follows: CanvasContext.setFontSize(number fontSize)

Function Description: Set the font size.

Parameters and Description: number fontSize, the size of the font.

Example Code

```
const ctx = wx.createCanvasContext('myCanvas')

ctx.setFontSize(20)
ctx.fillText('20', 20, 20)
ctx.setFontSize(30)
ctx.fillText('30', 40, 40)
ctx.setFontSize(40)
ctx.fillText('40', 60, 60)
ctx.setFontSize(50)
ctx.fillText('50', 90, 90)

ctx.draw()
```

©2013-2025 Tencent Cloud International Pte. Ltd.

setTextAlign

The method is used as follows: CanvasContext.setTextAlign(string align)

Function Description: Set the alignment of the text.

Parameters and Description: string align, the alignment of the text.

Value	Note
left	Align Left
center	Align Center
right	Align Right

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')

ctx.setStrokeStyle('red')
ctx.moveTo(150, 20)
ctx.lineTo(150, 170)
ctx.stroke()

ctx.setFontSize(15)
ctx.setTextAlign('left')
ctx.fillText('textAlign=left', 150, 60)

ctx.setTextAlign('center')
ctx.fillText('textAlign=center', 150, 80)

ctx.setTextAlign('right')
ctx.fillText('textAlign=right', 150, 100)

ctx.draw()
```

©2013-2025 Tencent Cloud International Pte. Ltd.

setTextBaseline

The method is used as follows: CanvasContext.setTextBaseline(string textBaseline)

Function Description: Sets the vertical alignment of the text.

Parameters and Description: string textBaseline, the vertical alignment method of the text.

Value	Note
top	Align Top
bottom	Align Bottom
middle	Align Center
normal	Default Value

Sample Code:

```
const ctx = wx.createCanvasContext('myCanvas')

ctx.setStrokeStyle('red')
ctx.moveTo(5, 75)
ctx.lineTo(295, 75)
ctx.stroke()

ctx.setFontSize(20)

ctx.setTextBaseline('top')
ctx.fillText('top', 5, 75)

ctx.setTextBaseline('middle')
ctx.fillText('middle', 50, 75)

ctx.setTextBaseline('bottom')
ctx.fillText('bottom', 120, 75)

ctx.setTextBaseline('normal')
ctx.fillText('normal', 200, 75)

ctx.draw()
```

©2013-2025 Tencent Cloud International Pte. Ltd.

©2013-2025 Tencent Cloud International Pte. Ltd.

Color

Last updated：2024-11-21 18:41:53

Colors. The colors used in canvas can be represented in several ways:

RGB colors: e.g. 'rgb(255, 0, 0)'.

RGBA colors: e.g. 'rgba(255, 0, 0, 0, 0.3)'

hexadecimal colors: e.g. '#FF0000'

Predefined colors: e.g. 'red'

There are 148 predefined colors: Note*: Color Name is case insensitive.

Color Name	HEX
AliceBlue	#F0F8FF
AntiqueWhite	#FAEBD7
Aqua	#00FFFF
Aquamarine	#7FFFD4
Azure	#F0FFFF
Beige	#F5F5
Bisque	#FFE4C4
Black	#000000
BlanchedAlmond	#FFEBCD
Blue	#0000FF
BlueViolet	#8A2BE2
Brown	#A52A2A
BurlyWood	#DEB887
CadetBlue	#5F9EA0
Chartreuse	#7FFF00
Chocolate	#D2691E
Coral	#FF7F50

©2013-2025 Tencent Cloud International Pte. Ltd.

CornflowerBlue	#6495ED
Cornsilk	#FFF8DC
Crimson	#DC143C
Cyan	#00FFFF
DarkBlue	#00008B
DarkCyan	#008B8B
DarkGoldenRod	#B8860B
DarkGray	#A9A9A9
DarkGrey	#A9A9A9
DarkGreen	#006400
DarkKhaki	#BDB76B
DarkMagenta	#8B008B
DarkOliveGreen	#556B2F
DarkOrange	#FF8C00
DarkOrchid	#9932CC
DarkRed	#8B0000
DarkSalmon	#E9967A
DarkSeaGreen	#8FBC8F
DarkSlateBlue	#483D8B
DarkSlateGray	#2F4F4F
DarkSlateGrey	#2F4F4F
DarkTurquoise	#00CED1
DarkViolet	#9400D3
DeepPink	#FF1493
DeepSkyBlue	#00BFFF

©2013-2025 Tencent Cloud International Pte. Ltd.

DimGray	#696969
DimGrey	#696969
DodgerBlue	#1E90FF
FireBrick	#B22222
FloralWhite	#FFFAF0
ForestGreen	#228B22
Fuchsia	#FF00FF
Gainsboro	#DCDCDC
GhostWhite	#F8F8FF
Gold	#FFD700
GoldenRod	#DAA520
Gray	#808080
Grey	#808080
Green	#008000
GreenYellow	#ADFF2F
HoneyDew	#F0FFF0
HotPink	#FF69B4
IndianRed	#CD5C5C
Indigo	#4B0082
Ivory	#FFFFF0
Khaki	#F0E68C
Lavender	#E6E6FA
LavenderBlush	#FFF0F5
LawnGreen	#7CFC00
LemonChiffon	#FFFACD

©2013-2025 Tencent Cloud International Pte. Ltd.

LightBlue	#ADD8E6
LightCoral	#F08080
LightCyan	#E0FFFF
LightGoldenRodYellow	#FAFAD2
LightGray	#D3D3D3
LightGrey	#D3D3D3
LightGreen	#90EE90
LightPink	#FFB6C1
LightSalmon	#FFA07A
LightSeaGreen	#20B2AA
LightSkyBlue	#87CEFA
LightSlateGray	#778899
LightSlateGrey	#778899
LightSteelBlue	#B0C4DE
LightYellow	#FFFFE0
Lime	#00FF00
LimeGreen	#32CD32
Linen	#FAF0E6
Magenta	#FF00FF
Maroon	#800000
MediumAquaMarine	#66CDAA
MediumBlue	#0000CD
MediumOrchid	#BA55D3
MediumPurple	#9370DB
MediumSeaGreen	#3CB371

©2013-2025 Tencent Cloud International Pte. Ltd.

MediumSlateBlue	#7B68EE
MediumSpringGreen	#00FA9A
MediumTurquoise	#48D1CC
MediumVioletRed	#C71585
MidnightBlue	#191970
MintCream	#F5FFFA
MistyRose	#FFE4E1
Moccasin	#FFE4B5
NavajoWhite	#FFDEAD
Navy	#000080
OldLace	#FDF5E6
Olive	#808000
OliveDrab	#6B8E23
Orange	#FFA500
OrangeRed	#FF4500
Orchid	#DA70D6
PaleGoldenRod	#EEE8AA
PaleGreen	#98FB98
PaleTurquoise	#AFEEEE
PaleVioletRed	#DB7093
PapayaWhip	#FFEFD5
PeachPuff	#FFDAB9
Peru	#CD853F
Pink	#FFC0CB
Plum	#DDA0DD

©2013-2025 Tencent Cloud International Pte. Ltd.

PowderBlue	#B0E0E6
Purple	#800080
RebeccaPurple	#663399
Red	#FF0000
RosyBrown	#BC8F8F
RoyalBlue	#4169E1
SaddleBrown	#8B4513
Salmon	#FA8072
SandyBrown	#F4A460
SeaGreen	#2E8B57
SeaShell	#FFF5EE
Sienna	#A0522D
Silver	#C0C0C0
SkyBlue	#87CEEB
SlateBlue	#6A5ACD
SlateGray	#708090
SlateGrey	#708090
Snow	#FFFAFA
SpringGreen	#00FF7F
SteelBlue	#4682B4
Tan	#D2B48C
Teal	#008080
Thistle	#D8BFD8
Tomato	#FF6347
Turquoise	#40E0D0

©2013-2025 Tencent Cloud International Pte. Ltd.

Violet	#EE82EE
Wheat	#F5DEB3
White	#FFFFFF
WhiteSmoke	#F5F5F5
Yellow	#FFFF00
YellowGreen	#9ACD32

©2013-2025 Tencent Cloud International Pte. Ltd.

Worker

Last updated：2024-11-21 18:41:53

wx.CreateWorker

wx.createWorker(string scriptPath)

Function: Create a worker thread.

Parameters: string scriptPath, the absolute path of the worker entry file.

Return：Worker， Worker object.

Notes:

Before using the interface, you need to configure the workers field in app.json, it means the root directory of the worker code;

scriptPath is the absolute path of the entry file, and does not start with /;

Currently, only one worker can be created at most, please calll Worker.terminate before creating the next worker.

Sample code

```
// Create a regular worker
const worker = wx.createWorker('workers/index.js') // The filename specifies the pa


worker.onMessage(function (res) {
  console.log(res)
})


worker.postMessage({
  msg: 'hello worker'
})


worker.terminate()
```

Worker

Worker.onMessage

Worker.onMessage(function listener)

Feature Description：Listen to the event of the message sent by the main thread/worker thread to the current thread.

Parameter 1：function listener, the event listener for the message sent by the main thread/worker thread to the current thread.

Parameter 2：**Object res**

Attributes	Types	Description
message	Object	Message sent from the main/Worker thread to the current thread

Worker.postMessage

Worker.postMessage(Object message)

Feature Description：**Message sent to the main/Worker thread.**

Parameters：**Object message, the message to be sent.**

Sample code:

worker thread

```
worker.postMessage({
  msg: 'hello from worker'
})
```

main thread

```
const worker = wx.createWorker('workers/request/index.js')

worker.postMessage({
  msg: 'hello from main'
})
```

Woker.terminate

Worker.terminate()

Feature Description：Ends the current Worker thread. Call only on the worker object of the main thread.

©2013-2025 Tencent Cloud International Pte. Ltd.

Media
Rich Text

Last updated：2025-03-20 18:01:56

EditorContext

Feature Description: An EditorContext instance can be obtained through wx.createSelectorQuery. The EditorContext is bound to an editor component through an id, thereby manipulating the corresponding editor component.

.blur

This method is used via EditorContext.blur(Object object).

Feature Description: The editor loses focus, concurrently retracting the keyboard.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.clear

This method is used via EditorContext.clear(Object object)

Feature Description: Empties the content of the editor.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description

success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.format

This method is used via EditorContext.format(string name, string value).

Feature Description: Modifies the style.

Parameter and Description:

string name, the attribute;

string value, value.

A list of styles that support the setting:

name	value
bold	-
italic	-
underline	-
strike	-
ins	-
script	sub / super
header	H1 / H2 / h3 / H4 / h5 / H6
align	left / center / right / justify
direction	rtl
indent	-1 / +1
list	ordered / bullet / check
color	hex color

backgroundColor	hex color
margin/marginTop/marginBottom/marginLeft/marginRight	css style
padding/paddingTop/paddingBottom/paddingLeft/paddingRight	css style
font/fontSize/fontStyle/fontVariant/fontWeight/fontFamily	css style
lineHeight	css style
letterSpacing	css style
textDecoration	css style
textIndent	css style
wordWrap	css style
wordBreak	css style
whiteSpace	css style

Setting a style on a selection that has already been styled removes the style. css style represents the allowed values specified in css.

.getContents

> This method is used via EditorContext.getContents().

Feature Description: Gets the content of the editor.
Parameter and Description: Object.

.getSelectionText

> This method is used via EditorContext.getSelectionText(Object object).

Feature Description: Gets the plain text content within the selected area of the editor. When the editor loses focus or a section is not selected, the returned content is empty.
Parameter and Description: Object.

Attribute	Type	Default value	Required	Description

success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success callback function

Arguments Object.res

Attribute	Type	Description
text	string	Plain text content

.insertDivider

This method is used via EditorContext.insertDivider(Object object)

Feature Description: Inserts a divider.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.insertImage

This method is used via EditorContext.insertImage(Object object).

Feature Description: Inserts an image.

When the address is a temporary file, the HTML format content of the editor obtained adds the attribute data-local to the tag, and the data-local field is added to the image attributes attribute in the data format content, which is the address of the temporary file passed in.
Developers can opt to upload images to the server during the submission phase, replacing them once the network address is obtained. For HTML content, the src value should be replaced, while for delta content, the insert {image: abc} value should be substituted.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
src	string	-	Yes	Image address, only http(s), base64, temp files are supported.
nowrap	boolean	false	No	Whether to automatically line feed after inserting a picture, default line feed
alt	string	-	No	Alternative text when images cannot be displayed
width	string	-	No	Image width (pixels/percentage)
height	string	-	No	Image height (pixels/percentage)
extClass	string	-	No	Class name added to the image img tag
data	Object	-	No	data is serialized as name=value;name1=value2 and hooked to the attribute data-custom.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Sample code

```
this.editorCtx.insertImage({
  src: 'xx',
  width: '100px',
  height: '50px',
  extClass: className
})
```

.insertText

This method is used via EditorContext.insertText(Object object).

Feature Description: Overwrites the current selected area, setting a segment of text.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
text	string	-	No	Text Content
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.redo

This method is used via EditorContext.redo(Object object).

Feature Description: Restoration.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.removeFormat

This method is used via EditorContext.removeFormat(Object object).

Feature Description: Clears the style of the current selected area.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.scrollIntoView

This method is used via EditorContext.scrollIntoView().

Feature Description: Enables the editor cursor to scroll within the visible area of the window.

.setContents

This method is used via EditorContext.setContents(Object object).

Feature Description: Initializes the editor content. When both html and delta exist, only delta takes effect.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
html	string	-	No	HTML content with tags
delta	Object	-	No	A delta object representing the content
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call

complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.undo

This method is used via EditorContext.undo(Object object).

Feature Description: Undo.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Video

Last updated：2025-06-09 10:04:51

saveVideoToPhotosAlbum

This API is called using wx.saveVideoToPhotosAlbum(Object object).

Feature description: Saves a video to the system album. Supported video format: MP4.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
filePath	string	-	True	The video file path, which can be a temporary file path or a permanent file path.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example

```
wx.saveVideoToPhotosAlbum({
  filePath: 'wxfile://xxx',
  success(res) {
    console.log(res.errMsg)
  },
})
```

createVideoContext

This API is called using VideoContext wx.createVideoContext(string id, Object this).

Feature description: Creates a VideoContext object.

Parameter and description:

string id: The ID of the <video> component.

In the context of a custom component, this refers to the current component instance, allowing you to manipulate the <video> component within it.

Return value: VideoContext

chooseVideo

This API is called using wx.chooseVideo(Object object).

Feature description: Records a video or selects a video from the phone's album.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
sourceType	Array.<string>	['album', 'camera']	False	The source options for video selection.
compressed	Boolean	true	False	Whether to compress the selected video file.
maxDuration	number	60	False	Maximum recording time for the video, in seconds.
camera	string	'back'	False	Default camera to open (front or rear). May not work on some Android devices due to system ROM limitations.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Valid values for object.sourceType

Value	Description
album	Selects a video from the album.
camera	Records a video using the camera.

Valid values for object.camera

Value	Description
back	Opens the rear camera by default.
front	Opens the front camera by default.

object.success callback function parameter: Object res.

Property	Type	Description
tempFilePath	string	The temporary file path of the selected video.
duration	number	The selected video duration in seconds.
size	number	The size of the selected video file, in bytes.
height	number	The height of the selected video.
width	number	The width of the selected video.

Example

```
wx.chooseVideo({
  sourceType: ['album', 'camera'],
  maxDuration: 60,
  camera: 'back',
  success(res) {
    console.log(res.tempFilePath)
  },
})
```

compressVideo

This API is called using wx.compressVideo(Object object).

Feature description: Compresses a video file. Developers can specify compression quality using `quality` . For more precise control, you can also specify `bitrate` , `fps` , and `resolution.` If `quality` is provided, these three parameters will be ignored.

Parameter and description: Object object.

Property	Type	Required	Description
src	string	True	The video file path, which can be a temporary file path or a permanent file path.
quality	string	True	Compression quality. Valid values: low: Low medium: Medium high: High
bitrate	number	True	Bitrate in kbps.
fps	number	True	Frame rate.
resolution.	number	True	The resolution ratio relative to the original video. Value range: (0, 1].
success	function	False	Callback function for successful API calls.
fail	function	False	Callback function for failed API calls.
complete	function	False	Callback function executed after API call ends (regardless of success or failure).

object.success callback function parameter: Object res

Property	Type	Description
tempFilePath	string	The temporary file path of the compressed video.
size	string	The size of the compressed video, in KB.

chooseMedia

This API is called using wx.chooseMedia(Object object).

Feature description: Takes or selects images or videos from the phone's album.

Parameter and description: Object object

Property	Type	Default value	Required	Description
count	number	15	False	Maximum number of files that can be selected, up to 15 files.
mediaType	Array.	['image', 'video']	False	File type. Valid values:

	<string>			image: Only allows taking photos or selecting photos from the album video: Only allows recording videos or selecting videos from the album **Note：** On iOS, any image format supported by the system photo app can be selected. Typically including JPG, PNG, HEIC (depending on device model and iOS version), and GIF.
sourceType	Array.<string>	['album', 'camera']	False	Source for selecting images and videos. Valid values: album: Selects from the album camera: Takes/records using the camera
maxDuration	number	10	False	Maximum recording time for videos, in seconds. The time range is between 3s and 60s. Videos selected from the album have no duration limit.
sizeType	Array.<string>	['original', 'compressed']	False	Effective only when mediaType is image. Indicates whether to compress the selected files.
camera	string	'back'	False	Effective only when sourceType is camera. Specifies whether to use the front or rear camera. back: Uses the rear camera front: Uses the front camera
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.success callback function parameter: Object res

Property	Type	Description
tempFiles	Array.<Object>	List of local temporary files.
type	string	File type. Valid values: image, video.

tempFiles structural property

Structural property	Type	Description
tempFilePath	string	Local temporary file path (local directory).
size	number	Size of the local temporary file in bytes.
duration	number	Video duration.
height	number	Video height.
width	number	Video width.
thumbTempFilePath	string	Temporary file path for video thumbnail.
fileType	string	File type. Valid values: image: Image video: Video

Example

```
wx.chooseMedia({
  count: 9,
  mediaType: ['image','video'],
  sourceType: ['album', 'camera'],
  maxDuration: 30,
  camera: 'back',
  success(res) {
    console.log(res.tempFiles.tempFilePath)
    console.log(res.tempFiles.size)
  }
})
```

VideoContext

Feature description:A VideoContext instance that can be obtained using wx.createVideoContext . The videoContext is bound to a <video> component via its ID, allowing you to manipulate the corresponding <video> component.

Example

Corresponding WXML file:

```
<view class="section tc">
  <video
    id="myVideo"
```

```
    src="http://wxsnsdy.tc.qq.com/105/20210/snsdyvideodownload?filekey=302802010104
    enable-danmu
    danmu-btn
    controls
  ></video>
  <view class="btn-area">
    <input bindblur="bindInputBlur" />
    <button bindtap="bindSendDanmu">Send on-screen comments</button>
  </view>
</view>
```

Corresponding js file:

```
function getRandomColor() {
  const 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(res) {
    this.videoContext = wx.createVideoContext('myVideo')
  },
  inputValue: '',
  bindInputBlur(e) {
    this.inputValue = e.detail.value
  },
  bindSendDanmu() {
    this.videoContext.sendDanmu({
      text: this.inputValue,
      color: getRandomColor(),
    })
  },
})
```

Methods

exitFullScreen

The method is called using VideoContext.exitFullScreen().

Feature description: Exits full screen.

exitBackgroundPlayback

The method is called using VideoContext.exitBackgroundPlayback().

Feature description: Exits background audio playback mode.

playbackRate

The method is called using VideoContext.playbackRate(number rate).

Feature description: Sets the playback speed.

Parameter and description:number rate, supports speeds of 0.5, 0.8, 1.0, 1.25, 1.5, and 2.0 times.

pause

The method is called using VideoContext.pause().

Feature description: Pauses the video.

play

The method is called using VideoContext.play().

Feature description: Plays the video.

requestFullScreen

The method is called using VideoContext.requestFullScreen(Object object).

Feature description: Enters full screen. If you have custom content that needs to be displayed in full screen, it should be placed within the video node.

Parameter and description: Object object

Property	Type	Default value	Required	Description

direction	number	-	False	Set the video orientation in full screen; if not specified, it will be automatically determined based on the aspect ratio.

Valid values for object.direction:

Value	Description
0	Normal portrait orientation.
90	Screen rotated 90 degrees counterclockwise.
-90	Screen rotated 90 degrees clockwise.

requestBackgroundPlayback

> The method is called using VideoContext.requestBackgroundPlayback().

Feature description: Enters background audio playback mode.

seek

> This method is called using VideoContext.seek(number position).

Feature description: Jumps to a specified position.

Parameter and description:number position, the position to jump to, in seconds.

sendDanmu

> The method is called using VideoContext.sendDanmu(Object data).

Feature description: Sends an on-screen comment.

Parameter: Object data. Content of the on-screen comment.

Property	Type	Default value	Required	Description
text	string	-	True	On-screen comment text.
color	string	-	False	On-screen comment color.

stop

> The method is called using VideoContext.stop().

Feature description: Stops the video.

Audio Recording

Last updated：2024-11-28 10:40:04

startRecord

This API is used via wx.startRecord(Object object).

Feature Description: Starts audio recording. The audio recording will automatically cease when wx.stopRecord is actively invoked, or when the audio recording exceeds one minute. This interface cannot be invoked when the user exits the mini program.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
tempFilePath	string	Temporary path of audio recording file (local path)

Sample Code

```
wx.startRecord({
  success (res) {
    const tempFilePath = res.tempFilePath
  }
})
setTimeout(function () {
  wx.stopRecord() // Stop audio recording
}, 10000)
```

©2013-2025 Tencent Cloud International Pte. Ltd.

stopRecord

This API is used via wx.stopRecord(Object object).

Feature Description: Stops audio recording.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Sample Code

```
wx.startRecord({
  success (res) {
    const tempFilePath = res.tempFilePath
  }
})
setTimeout(function () {
  wx.stopRecord() // Stop audio recording
}, 10000)
```

getRecorderManager

This API is used via RecorderManager wx.getRecorderManager().

Feature Description: Gets the globally unique RecorderManager.

Return Value: RecorderManager.

RecorderManager

.onError

This method is used via RecorderManager.onError(function listener).

Feature Description: Monitors audio recording error events.
Parameter and Description: function listener, the listener function for audio recording error events.

Attribute	Type	Description
errMsg	string	Error Message

.onFrameRecorded

This method is used via RecorderManager.onFrameRecorded(function listener).

Feature Description: Monitors events where files have been recorded to a specified frame size. If frameSize is set, this event will be called back.
Parameter and Description: function listener, the listener function for events where files have been recorded to a specified frame size.

Attribute	Type	Description
frameBuffer	ArrayBuffer	Audio recording shard data
isLastFrame	boolean	Is the current frame the last one before the normal end of the audio recording?

.onInterruptionBegin

This method is used via RecorderManager.onInterruptionBegin(function listener).

Feature Description: Monitors events where audio recording is interrupted due to system occupation. WeChat voice chat and WeChat video chat scenarios will trigger such events, and subsequently the audio recording will pause.
Parameter and Description: function listener, the listener function for events where audio recording is interrupted due to system occupation.

.onInterruptionEnd

This method is used via RecorderManager.onInterruptionEnd(function listener).

Feature Description: Monitors audio recording interruption ending events. After receiving the interruptionBegin event, all audio recordings within the mini program will pause, and only after receiving this event can audio recording be successfully resumed.
Parameter and Description: function listener, the listener function for audio recording interruption ending events.

.onPause

This method is used via RecorderManager.onPause(function listener).

Feature Description: Monitors audio recording pause events.
Parameter and Description: function listener, the listener function for audio recording pause events.

.onResume

This method is used via RecorderManager.onResume(function listener).

Feature Description: Monitors audio recording continuation events.
Parameter and Description: function listener, the listener function for audio recording continuation events.

.onStart

This method is used via RecorderManager.start(Object object).

Feature Description: Monitors audio recording start events.
Parameter and Description: function listener, the listener function for audio recording start events.

.onStop

This method is used via RecorderManager.onStop(function listener).

Feature Description: Monitors audio recording ending events.

©2013-2025 Tencent Cloud International Pte. Ltd.

Parameters and Description: function listener, the listener function for audio recording ending events. Object res parameter.

Attribute	Type	Description
tempFilePath	string	Temporary path of audio recording file (local path)
duration	number	Total audio recording duration, in milliseconds.
fileSize	number	Audio recording file size, in bytes.

.pause

This method is used via RecorderManager.pause().

Feature Description: Pauses audio recording.

.resume

This method is used via RecorderManager.resume().

Feature Description: Continues audio recording.

.start

This method is used via RecorderManager.start(Object object).

Feature Description: Starts audio recording.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
duration	number	60000	No	The duration of the audio recording, measured in milliseconds (ms), with a maximum value of 600,000 (equivalent to 10 minutes).
sampleRate	number	80000	No	Sampling rate (not supported on PC)

numberOfChannels	number	2	No	Number of audio recording channels
encodeBitRate	number	48000	No	Encoding bitrate, with valid values provided in the table below.
format	string	aac	No	Audio Format
frameSize	number	-	No	Specify the frame size in kilobytes (KB). Upon passing in the frameSize, the content of the files recorded to the specified frame size will be called back. If not specified, no callback will occur. Currently, only mp3 and pcm formats are supported.
audioSource	string	auto	No	Specify the audio input source for audio recording.

object.numberOfChannels Legal values

Legal value	Description
1	1 channel
2	2 channel

object.format Legal values

Legal value	Description
mp3	mp3
aac	aac
wav	wav
PCM	pcm

object.audioSource Legal values

Legal value	Description
auto	Automatically set up, the default use of cell phone microphone, plug in the headset automatically switch to use the headset microphone, all platforms applicable
buildInMic	Cellular microphone, iOS only
headsetMic	Wired headset microphone, iOS only

mic	Microphone (phone microphone when headset is not plugged in, headset microphone when headset is plugged in), Android only
camcorder	Same as mic for recording audio and video content, Android only
voice_communication	Same as mic, for real-time communication, Android only
voice_recognition	Same as mic, for speech recognition, Android only

Sampling rate and encoding bitrate limits

Each sampling rate corresponds to a valid range of encoding bitrates. Setting an invalid sampling rate or encoding bitrate will result in audio recording failure. The specific correspondences are as follows in the table.

Sampling rate.	Encoding bitrate
8000	16000 ~ 48000
11025	16000 ~ 48000
12000	24000 ~ 64000
16000	24000 ~ 96000
22050	32000 ~ 128000
24000	32000 ~ 128000
32000	48000 ~ 192000
44100	64000 ~ 320000
48000	64000 ~ 320000

.stop

This method is used via RecorderManager.stop().

Feature Description: Stop audio recording.

Background Audio

Last updated：2024-11-28 10:40:29

stopBackgroundAudio

The method of utilizing this API is wx.stopBackgroundAudio(Object object).

Function Description: Halts the playback of music.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

seekBackgroundAudio

The method for utilizing this API is wx.seekBackgroundAudio(Object object)

Function Description: Controls the progression of music playback.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
position	number	-	Supported	Music position, measured in seconds
success	function	-	Not required	Callback function for successful interface invocation

fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Example Code

```
wx.seekBackgroundAudio({
    position: 30
})
```

playBackgroundAudio

The method for employing this API is wx.playBackgroundAudio(Object object)

Function Description: Utilizes the background player for music playback. For the host client, only one background music can be played at a time. When the user exits the mini program, the music will pause; if the music player is occupied by another mini program, the music within the original mini program will cease to play.
Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
dataUrl	string	-	Supported	Music link, currently supporting formats include m4a, aac, mp3, wav
title	string	-	Not required	Music Title
coverImgUrl	string	-	Not required	Cover URL
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful

				invocations)

Example Code

```
wx.playBackgroundAudio({
  dataUrl: '',
  title: '',
  coverImgUrl: ''
})
```

pauseBackgroundAudio

The method for utilizing this API is wx.pauseBackgroundAudio(Object object)

Function Description: Pauses music playback.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

onBackgroundAudioStop

The method for utilizing this API is wx.onBackgroundAudioStop(function listener)

Function Description: Monitors music stop events.

Parameters and Description: function listener, the monitoring function for music stop events.

©2013-2025 Tencent Cloud International Pte. Ltd.

onBackgroundAudioPlay

The method for utilizing this API is wx.onBackgroundAudioPlay(function listener)

Function Description: Monitors music playback events.

Parameters and Description: function listener, the monitoring function for music playback events.

onBackgroundAudioPause

The method for utilizing this API is wx.onBackgroundAudioPause(function listener)

Function Description: Monitors music pause events.

Parameters and Description: function listener, the monitoring function for music pause events.

getBackgroundAudioPlayerState

The method for utilizing this API is wx.getBackgroundAudioPlayerState(Object object)

Function Description: Retrieves the status of background music playback.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success Callback Function Parameters: Object res.

Attributes	Types	Note
duration	number	The length of the selected audio (in seconds), returned only during music playback.
currentPosition	number	Selects the playback position of the audio (in s), only returned when the music is playing
status	number	Playback status, valid values: 0: Paused 1: Playing 2: No music playing
downloadPercent	number	The percentage of audio download progress, returned only during music playback.
dataUrl	string	Song data link, returned only during music playback.

Example Code

```
wx.getBackgroundAudioPlayerState({
  success (res) {
    const status = res.status
    const dataUrl = res.dataUrl
    const currentPosition = res.currentPosition
    const duration = res.duration
    const downloadPercent = res.downloadPercent
  }
})
```

getBackgroundAudioManager

Function Description: Acquire the globally unique background audio manager. When the mini program switches to the background, if the audio is playing, it can continue to play. However, the background status cannot manipulate the audio playback status through API calls.

BackgroundAudioManager

BackgroundAudioManager instance, can be obtained through wx.getBackgroundAudioManager.

Attributes

Types	Note
string src	The data source of the audio. Default is empty string, when new src is set, it will start playing automatically, currently supported formats are m4a, aac, mp3, wav
number startTime	The position where the audio starts playing, unit: s.
string title	Audio title, used for the native audio player's audio title (required). The share function in the native audio player, the title of the shared card, will also use this value.
string epname	Album name, the share function in the native audio player, the description of the shared card, will also use this value.
string singer	Singer's name, the share function in the native audio player, the description of the shared card, will also use this value.
string coverImgUrl	Cover image URL, used as the background image for the native audio player. The share function in the native audio player, the accompanying image and background of the shared card will also use this image.
string webUrl	Page link, the share function in the native audio player, the description of the shared card, will also use this value.
string protocol	Audio protocol. The default value is 'http', setting 'hls' can support the playback of live audio using the HLS protocol.
number playbackRate	Playback speed. Range 0.5-2.0, default is 1 (Android requires version 6 or above).
number duration	The length of the current audio (unit: s), only returned when there is a valid src (read-only).
number currentTime	The current playback position of the audio (unit: s), only returned when there is a valid src (read-only).
boolean paused	Whether it is currently paused or stopped (read-only).
number buffered	The time the audio has been buffered, it only guarantees that the content from the current playback time to this point has been buffered (read-only).

Method Set

onCanplay

©2013-2025 Tencent Cloud International Pte. Ltd.

The usage of this method is BackgroundAudioManager.onCanplay(function listener).

Function Description: Listens for the event when the background audio enters a playable state. However, it does not guarantee smooth playback afterwards.
Parameters and Description: function listener, the listening function for the event when the background audio enters a playable state.
onEnded

The usage of this method is BackgroundAudioManager.onEnded(function listener).

Function Description: Listens for the event when the background audio ends naturally.
Parameters and Description: function listener, the listening function for the event when the background audio ends naturally.
onError

The usage of this method is BackgroundAudioManager.onError(function listener).

Function Description: Listens for the event when the background audio playback encounters an error.
Parameters and Description: function listener, the listening function for the event when the background audio playback encounters an error.
onNext

The usage of this method is BackgroundAudioManager.onNext(function listener).

Function Description: Listens for the event when the user clicks on the next track in the system music playback panel (iOS only).
Parameters and Description: function listener, the listening function for the event when the user clicks on the next track in the system music playback panel.
onPause

The usage of this method is BackgroundAudioManager.onPause(function listener).

©2013-2025 Tencent Cloud International Pte. Ltd.

Function Description: Listens for the event when the background audio is paused.
Parameters and Description: function listener, the listening function for the event when the background audio is paused.

onPlay

> The usage of this method is BackgroundAudioManager.onPlay(function listener).

Function Description: Listens for the event when the background audio is played.
Parameters and Description: function listener, the listening function for the event when the background audio is played.

onPrev

> The usage of this method is BackgroundAudioManager.onPrev(function listener).

Function Description: Listens for the event when the user clicks on the previous track in the system music playback panel (iOS only).
Parameters and Description: function listener, the listening function for the event when the user clicks on the previous track in the system music playback panel.

onSeeked

> The usage of this method is BackgroundAudioManager.onSeeked(function listener).

Function Description: Listens for the event when the background audio completes the jump operation.
Parameters and Description: function listener, the listening function for the event when the background audio completes the jump operation.

onSeeking

> The usage of this method is BackgroundAudioManager.onSeeking(function listener).

Function Description: Listens for the event when the background audio begins the jump operation.
Parameters and Description: function listener, the listening function for the event when the background audio begins the jump operation.

onStop

©2013-2025 Tencent Cloud International Pte. Ltd.

The usage of this method is BackgroundAudioManager.onStop(function listener).

Function Description: Listens for the background audio stop event.
Parameters and Description: function listener, the listening function for the background audio stop event.
onTimeUpdate

The usage of this method is BackgroundAudioManager.onTimeUpdate(function listener).

Function Description: Listens for the background audio playback progress update event, only callbacks when the mini program is in the foreground.
Parameters and Description: function listener, the listening function for the background audio playback progress update event.
onWaiting

The usage of this method is BackgroundAudioManager.onWaiting(function listener).

Function Description: Listens for the audio loading event. It is triggered when the audio needs to stop and load due to insufficient data.
Parameters and Description: function listener, the listening function for the audio loading event.
pause

The usage of this method is BackgroundAudioManager.pause().

Function Description: Pauses the music.
Error Code

Error Codes	Error Message	Note
10001	-	System error
10002	-	Network Error
10003	-	File error, please verify if the response header is missing the Content-Length.

10004	-	Invalid format
-1	-	An unknown error occurred.

play

The method is utilized as BackgroundAudioManager.play().

Function Description: Plays music.

seek

The method is employed as BackgroundAudioManager.seek(number currentTime).

Function Description: Navigate to a specified location.

Parameters and Description: number currentTime, the position to navigate to, measured in seconds. Accurate to three decimal places, thus supporting millisecond-level precision.

Error Code

Error Codes	Error Message	Note
10001	-	System error
10002	-	Network Error
10003	-	File error, please verify if the response header is missing the Content-Length.
10004	-	Invalid format
-1	-	An unknown error occurred.

stop

The method is implemented as BackgroundAudioManager.stop().

Function Description: Halts music.

Error Code

Error Codes	Error Message	Note
10001	-	System error
10002	-	Network Error
10003	-	File error, please verify if the response header is missing the Content-Length.
10004	-	Invalid format
-1	-	An unknown error occurred.

Sample code

```
const backgroundAudioManager = wx.getBackgroundAudioManager()

backgroundAudioManager.title = 'This Very Moment'
backgroundAudioManager.epname = 'This Very Moment'
backgroundAudioManager.singer = 'Xu Wei'
backgroundAudioManager.coverImgUrl = 'http://y.gtimg.cn/music/photo_new/T002R300x30
// After setting the src, it will play automatically.
backgroundAudioManager.src = 'http://ws.stream.qqmusic.qq.com/M500001VfvsJ21xFqb.mp
```

Map

Last updated：2025-03-25 18:15:56

createMapContext

This API is used via wx.createMapContext(string mapId, Object this).

Feature Description: Creates a MapContext object.
Parameter and Description: string mapId, ID of the map component; Object this, "this" of the current component instance under custom components, used to operate the map component within.
Return Value: MapContext.

MapContext

Note:
Unless otherwise specified, the support level for MapContext methods by mobile manufacturers is assumed to be:
System Map (only supported by iOS): Yes;
Google Map (Supported by Android, IDE): Yes;
Huawei Map (only supported by Android): Yes;
Tencent Map: Yes;
Baidu Map: Yes;
Amap: Yes.
Exclusive for China：Tencent Map、Baidu Map.
MapContext instance, obtainable via wx.createMapContext.
MapContext is bound to a map component via an id, thereby manipulating the corresponding <map> component.
Sample Code

```
<!-- map.wxml -->
<map id="myMap" show-location />

<button type="primary" bindtap="getCenterLocation">Get location</button>
<button type="primary" bindtap="moveToLocation">Change location</button>
<button type="primary" bindtap="translateMarker">Move Marker</button>
<button type="primary" bindtap="includePoints">Scale the field of view to display a
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
// map.js
Page({
  onReady: function (e) {
    // Use wx.createMapContext to get map context.
    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()
  },
  translateMarker: function() {
    this.mapCtx.translateMarker({
      markerId: 0,
      autoRotate: true,
      duration: 1000,
      destination: {
        latitude:23.10229,
        longitude:113.3345211,
      },
      animationEnd() {
        console.log('animation end')
      }
    })
  },
  includePoints: function() {
    this.mapCtx.includePoints({
      padding: [10],
      points: [{
        latitude:23.10229,
        longitude:113.3345211,
      }, {
        latitude:23.00229,
        longitude:113.3345211,
      }]
    })
  }
})
```

.addArc

The method is used via MapContext.addArc(Object object).

Note：

Support: Not supported.

Feature Description: An arc line is added, and either a waypoint or an angle must be set. The waypoint must be within the valid coordinate range of the start and end points, otherwise a correct arc line cannot be generated. When setting the angle, the angle is prioritized. The angle is defined as the angle rotated counterclockwise from the starting point to the end point, tangent to the starting point. This feature is temporarily unsupported on the tool side.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
id	number	-	Yes	Arc ID
start	Object	-	Yes	Start point
end	Object	-	Yes	End point
pass	Object	-	No	Waypoint
angle	number	0	No	Included Angle
width	number	5	No	Line Width
color	number	#000000	No	Color of the line
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Structure attributes of "start"

Structure attributes	Type	Default value	Required	Description
longitude	number	-	Yes	Longitude

latitude	number	-	Yes	Latitude

Structure attributes of "end"

Structure attributes	Type	Default value	Required	Description
longitude	number	-	Yes	Longitude
latitude	number	-	Yes	Latitude

Structure attributes of "pass"

Structure attributes	Type	Default value	Required	Description
longitude	number	-	Yes	Longitude
latitude	number	-	Yes	Latitude

.addCustomLayer

The method is used via : MapContext.addCustomLayer(Object object)

Note：

Support: Not supported.

Feature Description: Adds a personalized layer.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
layerId	string	-	Yes	ID of personalized layer.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.addGroundOverlay

The method is used via MapContext.addGroundOverlay(Object object).

Note：

Support: supported by IDE, not supported by Android and ios.

Feature Description: Creates a custom image layer, which scales in accordance with the map's zoom level.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
id	String	-	Yes	ID of image layer
src	String	-	Yes	Image path, supporting network images, temporary paths, and code package paths.
bounds	Object	-	Yes	Longitude and latitude range covered by the image
visible	Boolean	true	No	Visible or not
zIndex	Number	1	No	Layer Rendering Sequence
opacity	Number	1	No	Layer Transparency
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Structure attributes of "bounds"

Structure attributes	Type	Default value	Required	Description
southwest	Object	-	Yes	Southwest corner's longitude and latitude
northeast	Object	-	Yes	Northeast corner's longitude and latitude

Structure attributes of "southwest"

Structure attributes	Type	Default value	Required	Description
longitude	number	-	Yes	Longitude
latitude	number	-	Yes	Latitude

Structure attributes of "northeast"

Structure attributes	Type	Default value	Required	Description
longitude	number	-	Yes	Longitude
latitude	number	-	Yes	Latitude

.addMarkers

The method is used via MapContext.addMarkers(Object object).

Feature Description: Adds a marker.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
markers	array	-	Yes	Same as the marker attribute passed into the map component.
clear	boolean	false	No	Whether to clear all markers on the map first.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.eraseLines

The method is used via MapContext.eraseLines(Object object).

Feature Description: Erases or grays out the lines that have been added to the map.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
lines	Array.<Object>	-	Yes	Array of line segments to be erased. See polyline attribute for details.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Structure attributes of "lines"

Structure attributes	Type	Default value	Required	Description
id	number	-	Yes	Line segment ID
index	number	-	Yes	Specifies a section of the line segment, with the starting point index of the line being 0.
point	Object	-	Yes	Specifies a point within a section of the line segment.
clear	boolean	true	No	True for erasure, false for graying out.

Structural attributes of a "point".

Structure attributes	Type	Default value	Required	Description
longitude	number	-	Yes	Longitude
latitude	number	-	Yes	Latitude

.fromScreenLocation

The method is used via MapContext.fromScreenLocation(Object object).

Feature Description: Gets the corresponding latitude and longitude of a point on the screen, with the coordinate origin at the top left corner of the map.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
x	number	-	Yes	X-coordinate value.
y	number	-	Yes	Y-coordinate value.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
latitude	number	Latitude
longitude	number	Longitude

.getCenterLocation

This method is used via MapContext.getCenterLocation(Object object).

Feature Description: Gets the latitude and longitude of the current map center. The return is in the gcj02 coordinate system, which can be used for wx.openLocation().

Parameter and Description: Object

Attribute	Type	Default value	Required	Description

iconPath	string	-	No	Image path, supporting network paths, local paths, and code package paths.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
longitude	number	Longitude
latitude	number	Latitude

.getRegion

This method is used via MapContext.getRegion(Object object).

Feature Description: Gets the field of view range of the current map.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
southwest	object	Southwest corner's longitude and latitude

northeast	object	Northeast corner's longitude and latitude

Structure attributes of "southwest"

Structure attributes	Type	Description
longitude	number	Longitude
latitude	number	Latitude

Structure attributes of "northeast"

Structure attributes	Type	Description
longitude	number	Longitude
latitude	number	Latitude

.getRotate

Note：

Not supported by iOS.

> This method is used via MapContext.getRotate(Object object).

Feature Description: Gets the rotation angle of the current map.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description

rotate	number	Rotation Angle

.getScale

This method is used via MapContext.getScale(Object object).

Feature Description: Gets the scaling level of the current map.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
scale	number	Scaling value

.getSkew

Note：

Not supported by iOS.

This method is used via MapContext.getSkew(Object object).

Feature Description: Gets the tilt angle of the current map.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call

fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
skew	number	Tilt Angle

.includePoints

Note：

Not supported by iOS.

> The method is used via MapContext.includePoints(Object object).

Feature Description: Scales the view field to display all latitudes and longitudes.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
points	array.	-	Yes	List of coordinate points to be displayed within the visible area.
padding	array.	-	No	The distance from the edges of the rectangle formed by the coordinate points to the edge of the map, measured in pixels. The format is [top, right, bottom, left]. On Android, only the first item in the array can be recognized, and the padding for top, bottom, left, and right is consistent. The developer tool does not currently support the padding parameter.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Structure attributes of "points"

Structure attributes	Type	Default value	Required	Description
longitude	number	-	Yes	Longitude
latitude	number	-	Yes	Latitude

.initMarkerCluster

The method is used via MapContext.initMarkerCluster(Object object).

Feature Description: Initializes the configuration for point clustering. The default configuration is used if this feature is not called.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
enableDefaultStyle	boolean	true	No	Enable the default clustering style.
zoomOnClick	boolean	true	No	Whether to implement cluster separation when a clustered marker point is clicked.
gridSize	number	60	No	The aggregatable distance (measured in pixels) for the clustering algorithm, that is, points closer than this value will be clustered together.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.moveAlong

This method is used via MapContext.moveAlong(Object object).

Feature Description: Moves the **marker** along a specified path, suitable for scenarios such as trajectory playback. A callback event is triggered upon animation completion. If the animation is in progress and the **moveAlong** method is called again for the same **marker**, the previous animation will be interrupted.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
markerId	number	-	Yes	Specify a marker.
path	array	-	Yes	The coordinate string of the moving path, with the coordinate point format being `{longitude, latitude}`.
autoRotate	boolean	true	No	Automatically adjust the rotation angle of the marker based on the path direction.
duration	number	-	Yes	The duration of smooth movement.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.moveToLocation

This method is used via MapContext.moveToLocation(Object object).

Feature Description: Moves the map center to the current location point, when the map component "show-location" must be set to true.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
longitude	number	-	No	Longitude
latitude	number	-	No	Latitude
success	function	-	No	Callback Function of Successful Interface Call

fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.on

This method is used via MapContext.on(string event, function callback).

Feature Description: Monitors map events.

Parameter 1: visualLayerEvent, a unified callback exit for the visual layer, supported from version 1.5.16.

Parameter	Type	Description
layerId	String	Layer ID
eventType	String	Event type
eventInfo	String	Event information

Parameter 2: markerClusterCreate, triggered when a new cluster is generated due to scaling or dragging, only returning information about the newly created cluster.

Parameter	Type	Description
clusters	Array<ClusterInfo>	Cluster data

Parameter 3: markerClusterClick, click events of the cluster.

Parameter	Type	Description
cluster	ClusterInfo	Cluster

ClusterInfo Structure

Parameter	Type	Description
clusterId	Number	Cluster ID
center	LatLng	Cluster coordinates
markerIds	Array<Number>	Array of point marker data within the aggregation cluster

Parameter 4: string event, the name of the event; function callback, the callback function for the event.

The valid values for the event include markerClusterCreate, markerClusterClick, and visualLayerEvent.

Sample Code

```
MapContext.on('visualLayerEvent', (res) => {})
MapContext.on('markerClusterCreate', (res) => {})
MapContext.on('markerClusterClick', (res) => {})
```

.openMapApp

This method is used via MapContext.openMapApp(Object object).

Feature Description: Launches the map App for navigation.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
longitude	number	-	Yes	Destination Longitude
latitude	number	-	Yes	Destination Latitude
destination	string	-	Yes	Destination Name
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.removeArc

This method is used via MapContext.removeArc(Object object).

Note：

Support: Not supported.

Feature Description: Removes the arc line.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
id	number	-	Yes	Arc ID
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.removeCustomLayer

The method is used via MapContext.removeCustomLayer(Object object).

Note：

Support: Not supported.

Feature Description: Removes personalized image layers.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
layerId	string	-	Yes	ID of personalized layer.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.removeGroundOverlay

This method is used via MapContext.removeGroundOverlay(Object object).

©2013-2025 Tencent Cloud International Pte. Ltd.

Note：

Support: supported by IDE, not supported by Android and ios.

Feature Description: Removes custom image layers.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
id	String	-	Yes	ID of image layer
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.removeMarkers

This method is used via MapContext.removeMarkers(Object object).

Feature Description: Removes markers.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
markerIds	array	-	Yes	Collection of marker IDs.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.setBoundary

This method is used via MapContext.setBoundary(Object object).

Feature Description: Restricts the map's display range. This interface will also limit the map's minimum scaling integer level.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
southwest	Object	-	Yes	Southwest corner's longitude and latitude
northeast	Object	-	Yes	Northeast corner's longitude and latitude
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Structure attributes of "southwest"

Structure attributes	Type	Default value	Required	Description
longitude	number	-	Yes	Longitude
latitude	number	-	Yes	Latitude

Structure attributes of "northeast"

Structure attributes	Type	Default value	Required	Description
longitude	number	-	Yes	Longitude
latitude	number	-	Yes	Latitude

.setCenterOffset

This method is used via MapContext.setCenterOffset(Object object).

Feature Description: Sets the offset of the map center point, with growth towards the back and downwards, within a screen ratio range of (0.25 - 0.75), default offset being [0.5, 0.5].

©2013-2025 Tencent Cloud International Pte. Ltd.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
offset	Array.<number>	-	Yes	Offset value, a two-element array.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.setLocMarkerIcon

This method is used via MapContext.setLocMarkerIcon(Object object).

Feature Description: Sets the location point icon, supporting network paths, local paths, and code package paths.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
iconPath	string	-	No	Image path, supporting network paths, local paths, and code package paths.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.toScreenLocation

The method is used via MapContext.toScreenLocation(Object object).

©2013-2025 Tencent Cloud International Pte. Ltd.

Feature Description: Gets the screen coordinates corresponding to the latitude and longitude, with the coordinate origin at the top left corner of the map.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
latitude	Number	-	Yes	Latitude
longitude	Number	-	Yes	Longitude
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
x	number	X-coordinate value.
y	number	Y-coordinate value.

.translateMarker

The method is used via MapContext.translateMarker(Object object).

Feature Description: Translates the marker, with iOS and IDE supporting animation effects. Android platform does not currently support animation effects.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
markerId	number	-	Yes	Specify a marker.
destination	object	-	Yes	Specify the target point for the marker to move to.
animationEnd	function	-	No	Animation completion callback function

©2013-2025 Tencent Cloud International Pte. Ltd.

success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Structure attributes of "destination"

Structure attributes	Type	Default value	Required	Description
longitude	number	-	Yes	Longitude
latitude	number	-	Yes	Latitude

.updateGroundOverlay

This method is used via MapContext.updateGroundOverlay(Object object).

Note：

Support: supported by IDE, not supported by Android and ios.

Feature Description: Updates custom image layers.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
id	String	-	Yes	ID of image layer
src	String	-	Yes	Image path, supporting network images, temporary paths, and code package paths.
bounds	Object	-	Yes	Longitude and latitude range covered by the image
visible	Boolean	true	No	Visible or not
zIndex	Number	1	No	Layer Rendering Sequence
opacity	Number	1	No	Layer Transparency
success	function	-	No	Callback Function of Successful Interface Call

fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Structure attributes of "bounds"

Structure attributes	Type	Default value	Required	Description
southwest	Object	-	Yes	Southwest corner's longitude and latitude
northeast	Object	-	Yes	Northeast corner's longitude and latitude

Structure attributes of "southwest"

Structure attributes	Type	Default value	Required	Description
longitude	number	-	Yes	Longitude
latitude	number	-	Yes	Latitude

Structure attributes of "northeast"

Structure attributes	Type	Default value	Required	Description
longitude	number	-	Yes	Longitude
latitude	number	-	Yes	Latitude

©2013-2025 Tencent Cloud International Pte. Ltd.

Real-Time Audio/Video

Last updated：2025-03-25 18:15:55

createLivePusherContext

This method is used via LivePusherContext wx.createLivePusherContext().

Note:

Support for this feature starts from the base library 1.4.96. Compatibility measures are required for earlier versions.

Feature Description: Creates a LivePusherContext object.

Return Value: LivePusherContext.

LivePusherContext

LivePusherContext instance, obtainable via wx.createLivePusherContext.

The LivePusherContext is uniquely bound within the page to the live-pusher component, operating the corresponding live-pusher component.

.start

This method is used via LivePusherContext.start(Object object).

Feature Description: Starts streaming while simultaneously activating camera preview.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.stop

This method is used via LivePusherContext.stop(Object object).

Feature Description: Stops streaming and concurrently camera preview.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.pause

This method is used via LivePlayerContext.pause(Object object).

Feature Description: Pauses streaming.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.resume

This method is used via LivePusherContext.resume(Object object).

©2013-2025 Tencent Cloud International Pte. Ltd.

Feature Description: Resumes streaming.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.switchCamera

This method is used via LivePusherContext.switchCamera(Object object).

Feature Description: Switches between front and rear cameras.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.snapshot

This method is used via LivePusherContext.snapshot(Object object).

Feature Description: Snapshot.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description

success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success callback function: arguments Object res

Attribute	Type	Description
tempImagePath	string	Temporary path for image files
width	string	Width of the image
height	string	height of the image

.toggleTorch

This method is used via LivePusherContext.toggleTorch(Object object).

Feature Description: Switches to flashlight.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.playBGM

This method is used via LivePusherContext.playBGM(Object object).

Feature Description: Plays background sound.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
url	String	-	Yes	Resource address for adding background mix.
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.stopBGM

This method is used via LivePusherContext.stopBGM(Object object).

Feature Description: Stops background sound.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.pauseBGM

This method is used via LivePusherContext.pauseBGM(Object object).

Feature Description: Pauses background sound.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description

success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.resumeBGM

This method is used via LivePusherContext.resumeBGM(Object object).

Feature Description: Resumes background sound.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.setBGMPosition

This method is used via LivePusherContext.setBGMPosition(Object object).

Feature Description: Sets the progress of the background sound.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
position	Number	-	Yes	Background sound progress, measured in seconds.
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call

©2013-2025 Tencent Cloud International Pte. Ltd.

complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.setMICVolume

This method is used via LivePusherContext.setMICVolume(Object object).

Feature Description: Sets the microphone volume.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
volume	String	-	Yes	Volume level, ranging from 0 to 1.
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.setBGMVolume

This method is used via LivePusherContext.setBGMVolume(Object object).

Feature Description: Sets the background volume.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
volume	String	-	Yes	Volume level, ranging from 0 to 1.
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of

				the interface invocation (both successful and unsuccessful invocations)

.setAudioReverbType

This method is used via LivePusherContext.setAudioReverbType(Object object).

Feature Description: Sets the mix type.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
audioReverbType	Number	-	Yes	Mix type, 0 to 6 correspond to "Reverb off", "KTV", "Small room", "Large hall", "Deep", "Resonant", "Magnetic", respectively.
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.startPreview

This method is used via LivePusherContext.startPreview(Object object).

Feature Description: Enables camera preview.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and

©2013-2025 Tencent Cloud International Pte. Ltd.

				unsuccessful invocations)

.stopPreview

This method is used via LivePusherContext.stopPreview(Object object).

Feature Description: Disables camera preview.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.startAudioRecord

This method is used via LivePusherContext.startAudioRecord(Object object).

Feature Description: Starts audio recording, which will automatically terminate when LivePusherContext.stopAudioRecord is actively invoked, or when the recording exceeds one minute.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.stopAudioRecord

This method is used via LivePusherContext.stopAudioRecord(Object object).

Feature Description: Terminates audio recording.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
tempFilePath	String	Temporary path of audio recording file (local path)

createLivePlayerContext

This API is used via LivePlayerContext wx.createLivePlayerContext(string id, Object this).

Note:

Support for this feature starts from the base library 1.4.96. Compatibility measures are required for earlier versions.

Feature Description: Creates a LivePlayerContext object.

Parameter and Description:

string id, the ID of the live-player component;

Object this, "this" of the current component instance under custom components, used to operate the live-player component within.

Return Value: LivePlayerContext.

LivePlayerContext

.exitfullscreen

This method is used via LivePlayerContext.exitFullScreen(Object object).

Feature Description: Exits full screen.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.mute

This method is used via LivePlayerContext.mute(Object object).

Feature Description: Mute.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.pause

This method is used via LivePlayerContext.pause(Object object).

©2013-2025 Tencent Cloud International Pte. Ltd.

Feature Description: Pause.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.play

This method is used via LivePlayerContext.play(Object object).

Feature Description: Play.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.requestFullScreen

This method is used via LivePlayerContext.requestFullScreen(Object object).

Feature Description: Enters full screen.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description

direction	Number	0	No	Set the direction during full screen.
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Valid values of object.direction.

Value	Description
0	Normal vertical orientation
90	Screen rotating 90 degrees counterclockwise
-90	Screen rotating 90 degrees clockwise

.resume

This method is used via LivePlayerContext.resume(Object object).

Feature Description: Restoration.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.snapshot

This method is used via LivePlayerContext.snapshot(Object object).

Feature Description: Screenshot.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
tempImagePath	String	Temporary path of the image file
width	String	Image width
height	String	Image height

.stop

This method is used via LivePlayerContext.stop(Object object).

Feature Description: Halt.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	Function	-	No	Callback Function of Successful Interface Call
fail	Function	-	No	Callback Function of Failing Interface Call
complete	Function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

©2013-2025 Tencent Cloud International Pte. Ltd.

Transparent Video

Last updated：2025-03-25 18:15:55

createAnimationVideoContext

This API is used via AnimationVideoContext wx.createAnimationVideo(string id, Object this)

Feature Description: Creates an AnimationVideoContext object.

Parameter and Description:

string id: The id of the <animation-video> component.

Object this, "this" of the current component instance under custom components, used to operate the <animation-video> component within.

Return Value: AnimationVideoContext.

AnimationVideoContext

Feature Description: AnimationVideoContext instance, which can be obtained through wx.createAnimationVideo. AnimationVideoContext is bound to an <animation-video> component through its id, thereby manipulating the corresponding <animation-video> component.

Sample Code

```
<view class="wrap">
  <view class="card-area">
    <view class="top-description border-bottom">
        <view>Feature</view>
        <view>play pause seek</view>
    </view>
    <view class="video-area">
        <animation-video
            id="myAnimationVideo"
            path="{{leftAlphaSrcPath}}"
            loop="{{loop}}"
            resource-width="800"
            resource-height="400"
            canvas-style="width:200px;height:200px"
            autoplay="{{autoplay}}"
            bindstarted="onStarted"
```

```
            bindended="onEnded"
        ></animation-video>
    </view>
    <button bindtap="play">
      Play
    </button>
    <button bindtap="pause">
      Pause
    </button>
    <button bindtap="seek">
      Navigate to the 2-second mark of the animation.
    </button>
  </view>
</view>
```

```
Page({
  onLoad() {
    this.createCtx();
  },
  createCtx() {
    this.myAnimationVideo = wx.createAnimationVideoContext('my-video');
  },
  play() {
    this.myAnimationVideo?.play();
  },
  pause() {
    this.myAnimationVideo?.pause();
  },
  seek() {
    this.myAnimationVideo?.seek(2);
  },
})
```

.play

This method is used via AnimationVideoContext.play().

Feature Description: Plays the video.

.pause

This method is used via AnimationVideoContext.pause().

©2013-2025 Tencent Cloud International Pte. Ltd.

Feature Description: Pauses the video.

.seek

This method is used via AnimationVideoContext.seek(number position).

Feature Description: Redirects to the specified position.

Parameter and Description: number position, the position to redirect to, unit: seconds.

Camera

Last updated：2025-06-03 18:00:36

createCameraContext

This method is used via CameraContext wx.createCameraContext().

Feature Description: Creates a CameraContext object.
Return Value: CameraContext.

CameraContext

Feature Description: A CameraContext instance, which can be obtained through wx.createCameraContext. The `cameraContext` is bound to the unique `<camera>` component within the page, operating the corresponding `<camera>` component.

.onCameraFrame

This method is used via CameraFrameListener CameraContext.onCameraFrame(onCameraFrameCallback callback).

Feature Description: Gets real-time frame data from the Camera.
Parameter and Description: Function callback.
Callback function parameter: Object res.

Attribute	Type	Description
width	number	The width of the image data rectangle.
height	number	The height of the source image data rectangle.
data	ArrayBuffer	Image pixel data, a one-dimensional array, with every four items representing the RGBA of a pixel.

Return Value:CameraFrameListener。

©2013-2025 Tencent Cloud International Pte. Ltd.

Sample Code

```
const context = wx.createCameraContext()
const listener = context.onCameraFrame((frame) => {
  console.log(frame.data instanceof ArrayBuffer, frame.width, frame.height)
})
listener.start()
```

.setZoom

This method is used via CameraContext.setZoom(Object object).

Feature Description: Sets the scaling level.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
zoom	number	-	Yes	Scaling level, ranging [1, maxZoom]. The zoom can be a decimal, accurate to one decimal place. maxZoom can be obtained from the return value of bindinitdone.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
zoom	number	The actual set scaling level. Due to system constraints, some models may not be able to be set to the specified value, and will instead use the closest feasible value.

.startRecord

This method is used via CameraContext.startRecord(Object object).

Feature Description: Starts video recording.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
timeoutCallback	function	-	No	The recording will terminate after 30 seconds or when the page `onHide` is triggered.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameter of the object.timeoutCallback callback function: Object res.

Attribute	Type	Description
tempThumbPath	string	Temporary path of the cover image file
tempVideoPath	string	Temporary path of the video file

.stopRecord

This method is used via CameraContext.startRecord(Object object).

Feature Description: Stops video recording.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
tempThumbPath	string	Temporary path of the cover image file
tempVideoPath	string	Temporary path of the video file

.takePhoto

This method is used via CameraFrameListener.stop(Object object).

Feature Description: Captures photographs.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
quality	string	normal	No	Image Quality
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Valid values of object.quality

Value	Description
high	Higher quality
normal	Standard Quality
low	Lower quality

Parameters for object.success callback function: Object res.

Attribute	Type	Description
tempImagePath	string	Temporary path (local path) for photo files, Android is jpg image format, ios is png

©2013-2025 Tencent Cloud International Pte. Ltd.

CameraFrameListener

Note:

Listener Returned by CameraContext.onCameraFrame()

For related documentation, see camera.

.start

This method is used via CameraFrameListener.start(Object object).

Feature Description: Starts monitoring frame data.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.stop

This method is used via CameraFrameListener.stop(Object object).

Feature Description: Stops monitoring frame data.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

©2013-2025 Tencent Cloud International Pte. Ltd.

©2013-2025 Tencent Cloud International Pte. Ltd.

Image

Last updated：2025-06-06 19:00:43

chooseImage

This API is called using wx.chooseImage(Object object).

Feature description: Allows users to select images from the local photo album or take photos using the camera.

Parameter and description: Object object.

Property	Type	Valid values and description	Default value	Required	Description
count	number	-	9	False	The maximum number of images that can be selected.
sizeType	Array.<string>	original: Full-size image compressed: Compressed image	['original', 'compressed']	False	The size of the selected images.
sourceType	Array.<string>	album: Select images from the album camera: Use the camera	['album', 'camera']	False	The source of the images.
success	function	-	-	False	Callback function for successful API calls.
fail	function	-	-	False	Callback function for failed API calls.
complete	function	-	-	False	Callback function executed after API call ends (regardless of success or failure)

object.success callback function parameter: Object res.

Property	Type	Description
tempFilePaths	Array.<string>	List of local temporary file paths for the images.
tempFiles	Array.<Object>	List of local temporary files for the images.

res.tempFiles structure

Property	Type	Description
path	string	Local temporary file path.
size	number	Size of the local temporary file in bytes.

Example:

```
wx.chooseImage({
  count: 1,
  sizeType: ['original', 'compressed'],
  sourceType: ['album', 'camera'],
  success (res) {
    // tempFilePath can be used as the src attribute of an img tag to display the i
    const tempFilePaths = res.tempFilePaths
  }
})
```

compressImage

This API is called using wx.compressImage(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Compresses an image with optional compression quality settings.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
src	string	-	True	The path of the image, supports local paths and

				code package paths.
quality	number	80	False	Compression quality ranges from 0 to 100. A lower value results in lower quality and higher compression rate (applicable only to JPG).
compressedWidth	number	-	False	The width of the compressed image in px. If not specified, it will scale proportionally based on compressedHeight.
compressedHeight	number	-	False	The height of the compressed image in px. If not specified, it will scale proportionally based on compressedWidth.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure)

object.success callback function parameter: Object res.

Property	Type	Description
tempFilePath	string	The local temporary file path of the compressed image.

Example:

```
wx.compressImage({
  src: '', // Image path
  quality: 80 // Compression quality
})
```

getImageInfo

This API is called using wx.getImageInfo(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Gets image information. For network images, the download domain must be configured first.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
src	string	-	True	The path of the image, which can be a relative path, temporary file path, storage file path, or network image path.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure)

object.success callback function parameter: Object res.

Property	Type	Description
width	number	The original width of the image in px. Rotation is not considered.
height	number	The original height of the image in px. Rotation is not considered.
path	string	The local path of the image.
orientation	string	The orientation of the device when the photo was taken.
type	string	Image format.

Valid values for res.orientation

Value	Description
up	Default orientation (phone held horizontally), corresponds to Exif 1, or no orientation information.
up-mirrored	Same as up, but mirrored, corresponds to Exif 2.
down	Rotates 180 degrees, corresponds to Exif 3.
down-mirrored	Same as down, but mirrored, corresponds to Exif 4.
left-mirrored	Same as left, but mirrored, corresponds to Exif 5.
right	Rotated 90 degrees clockwise, corresponds to Exif 6.
right-mirrored	Same as right, but mirrored, corresponds to Exif 7.
left	Rotated 90 degrees counterclockwise, corresponds to Exif 8.

©2013-2025 Tencent Cloud International Pte. Ltd.

Valid values for res.type

Valid value	Description
unknown	Unknown format.
jpeg	JPEG compressed format.
png	PNG compressed format.
gif	GIF compressed format.
tiff	TIFF compressed format.

Example:

```
wx.getImageInfo({
  src: 'images/a.jpg',
  success(res) {
    console.log(res.width)
    console.log(res.height)
  },
})

wx.chooseImage({
  success(res) {
    wx.getImageInfo({
      src: res.tempFilePaths[0],
      success(res) {
        console.log(res.width)
        console.log(res.height)
      },
    })
  },
})
```

previewImage

This API is called using wx.previewImage(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Allows users to preview images in full screen on a new page. During the preview, users can save the image, send it to friends, and perform other actions.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
urls	Array.<string>	-	True	List of image URLs to preview. Supports cloud file IDs.
current	string	The first URL	False	The URL of the currently displayed image.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure)

Example:

```
wx.previewImage({
  current: '', // The URL of the currently displayed image
  urls: [], // List of image URLs to preview
})
```

previewMedia

This API is called using wx.previewMedia(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Allows users to preview images and videos.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
sources	Array.<Object>	-	True	List of resources to preview.

current	number	0	False	The index of the currently displayed resource.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure)

saveImageToPhotosAlbum

This API is called using wx.saveImageToPhotosAlbum(Object object).

Note:

This API is supported in mini programs but not in mini games.

Requires user authorization for scope.writePhotosAlbum.

Feature description: Saves an image to the system album.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
filePath	string	-	True	Image file path, which can be a temporary file path or a permanent file path (local path). Network image paths are not supported.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure)

Example:

```
wx.saveImageToPhotosAlbum({
  success(res) {},
})
```

chooseMessageFile

This API is called using wx.chooseMessageFile(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Selects a file from the client session.

Parameter and description: Object object.

Property	Type	Valid values and description	Default value	Required	Description
count	number	-	-	True	The maximum number of files that can be selected. Value range: 0 to 100.
type	string	all: Select from all files video: Only select video files image: Only select image files file: Select other files excluding images and videos	'all'	False	Type of files to be selected.
extension	Array.<string>	-	-	False	Filters by file extension, valid only when type==file. Each item must not be an empty string.

					No filtering by default.
success	function	-	-	False	Callback function for successful API calls.
fail	function	-	-	False	Callback function for failed API calls.
complete	function	-	-	False	Callback function executed after API call ends (regardless of success or failure)

chooseExternalFile

This API is called using wx.chooseExternalFile(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Selects a file from the mobile file picker.

Parameter and description: Object object.

Property	Type	Valid values and description	Default value	Required	Description
count	number	-	-	True	The maximum number of files that can be selected. Value range: 0 to 100.
type	string	all: Select from	'all'	False	Type of files to

		all files video: Only select video files image: Only select image files file: Select other files excluding images and videos			be selected.
extension	Array.<string>	-	-	False	Filters by file extension, valid only when type==file. Each item must not be an empty string. No filtering by default.
success	function	-	-	False	Callback function for successful API calls.
fail	function	-	-	False	Callback function for failed API calls.
complete	function	-	-	False	Callback function executed after API call ends (regardless of success or failure)

Audio

Last updated：2025-05-26 10:48:33

stopVoice

This API is called using wx.stopVoice(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Stops the voice playback.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
wx.startRecord({
  success (res) {
    const tempFilePath = res.tempFilePath
    wx.playVoice({
      filePath: tempFilePath,
    })

    setTimeout(() => { wx.stopVoice() }, 5000)
  }
})
```

playVoice

This API is called using wx.playVoice(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Starts the voice playback. Only one voice file can be played at a time. If a previous voice file is still playing, it will be interrupted.

Parameter and description:

Property	Type	Default value	Required	Description
filePath	string	-	True	The file path of the voice file to be played (local path).
duration	number	60	False	Specifies the duration of playback in seconds. Playback will stop automatically after this duration.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
wx.startRecord({
  success (res) {
    const tempFilePath = res.tempFilePath
    wx.playVoice({
      filePath: tempFilePath,
      complete () { }
    })
  }
})
```

pauseVoice

This API is called using wx.pauseVoice(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Pauses the voice playback. When you call wx.playVoice to play the same file again, it starts from where the file was paused. If you want to play from the beginning, you first need to call wx.stopVoice.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
wx.startRecord({
  success (res) {
    const tempFilePath = res.tempFilePath
    wx.playVoice({
      filePath: tempFilePath
    })

    setTimeout(() => { wx.pauseVoice() }, 5000)
  }
})
```

setInnerAudioOption

This API is called using wx.setInnerAudioOption(Object object).

Feature description: Sets the playback options for InnerAudioContext, and applies globally to the current mini program.

Parameter: Object object.

Property	Type	Default value	Required	Description
mixWithOther	boolean	true	False	Whether to mix with other audio. If set to true, it won't

				stop other app's music.
obeyMuteSwitch	boolean	true	False	(iOS only) Whether to follow the mute switch. If set to false, it can play sound even in silent mode.
speakerOn	boolean	true	False	true means using the speaker, false means using the handset. Default is true.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

getAvailableAudioSources

Note:

This API is supported in mini programs but not in mini games.

This API is called using wx.getAvailableAudioSources(Object object).

Feature description: Gets the supported audio input sources.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.success callback function parameter: Object res.

Property	Type	Description
audioSources	Array.<string>	The list of supported audio input sources. For values, see AudioSource

AudioSources valid values

Value	Description
auto	Automatically set. Uses phone mic by default, switches to headset mic when plugged in. Applicable to all platforms.
buildInMic	Phone microphone, iOS only.
headsetMic	Headset microphone, iOS only.
mic	Microphone (phone mic when no headset, headset mic when plugged in), Android only.
camcorder	Same as mic, suitable for recording audio/video content, Android only.
voice_communication	Same as mic, suitable for real-time communication, Android only.
voice_recognition	Same as mic, suitable for voice recognition, Android only.

createAudioContext

This API is called using AudioContext wx.createAudioContext(string id, Object this).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Creates an AudioContext object.

Parameter and description: String id: The ID of the audio component. Object this: In custom components, this refers to the current component instance to operate on the audio component within.

Return value: AudioContext

AudioContext

Note:

This API is supported in mini programs but not in mini games.

An AudioContext instance that can be obtained using wx.createAudioContext. It binds to an audio component via its ID and allows operations on the corresponding audio component.

pause

This method is called using AudioContext.pause().

Feature description: Pauses the audio.

play

This method is called using AudioContext.play ().

Feature description: Plays the audio.

seek

This method is called using AudioContext.seek(number position).

Feature description: Jumps to the specified position in the audio.

Parameter and description: number position. The position to jump to, in seconds.

setSrc

This method is called using AudioContext.setSrc(string src).

Feature description: Sets the audio source URL.

Parameter and description:string src: The audio source URL.

createInnerAudioContext

This API is called using InnerAudioContext wx.createInnerAudioContext(Object object).

Feature description: Creates an internal `audio` context, i.e., an InnerAudioContext object.

Return value: InnerAudioContext.

Note: InnerAudioContext audio resources are not automatically released. If the audio is no longer needed, call InnerAudioContext.destroy() to release resources and avoid memory leaks.

Example:

```
const innerAudioContext = wx.createInnerAudioContext()
innerAudioContext.src = 'http://ws.stream.qqmusic.qq.com/M500001VfvsJ21xFqb.mp3?gui

innerAudioContext.play() // Play the audio

innerAudioContext.pause () // Pause the audio

innerAudioContext.stop () // Stop the audio

innerAudioContext.destroy () // Release audio resources
```

InnerAudioContext

Property

string src: The URL of the audio resource to play.
number startTime: The position to start playing from (in seconds), default is 0.
boolean autoplay: Whether to start playing automatically, default is false.
boolean loop: Whether to loop the audio, default is false.
number volume: The volume level, ranging from 0 to 1. Default is 1.
number duration: The length of the current audio (in seconds). Returns only if there is a valid src (read-only).
number currentTime: The current playback position (in seconds). Returns only if there is a valid src, with time accurate to 6 decimal places (read-only).
boolean paused: Whether the audio is currently paused or stopped (read-only).
number buffered: The time point up to which the audio has been buffered. Only guarantees that the content from the current playback time to this time point has been buffered (read-only).

Supported formats

Format	iOS	Android
flac	x	√
m4a	√	√
ogg	x	√
ape	x	√
amr	x	√
wma	x	√

wav	√	√
mp3	√	√
mp4	x	√
aac	√	√
aiff	√	x
caf	√	x

Example

```
const innerAudioContext = wx.createInnerAudioContext()
innerAudioContext.autoplay = true
innerAudioContext.src = 'https://ws.stream.qqmusic.qq.com/M500001VfvsJ21xFqb.mp3?gu
innerAudioContext.onPlay(() => {
  console.log('Starts to play')
})
innerAudioContext.onError((res) => {
  console.log(res.errMsg)
  console.log(res.errCode)
})
```

Methods collection

play

This method is called using InnerAudioContext.play ().

Feature description: Plays the audio.

pause

This method is called using InnerAudioContext.pause().

Feature description: Pauses the audio. When played again, it will resume from the paused position.

stop

This method is called using InnerAudioContext.stop().

Feature description: Stops the audio. When played again, it will start from the beginning.

seek

This method is called using InnerAudioContext.seek(number position).

Feature description: Jumps to the specified position in the audio.

destroy

This method is called using InnerAudioContext.destroy ().

Feature description: Terminates the current instance.

onCanplay

This method is called using InnerAudioContext.onCanplay(function listener).

Feature description: Listens for the event when the audio can start playing. This does not guarantee smooth playback.

Parameter and description:function listener.The listener for the event when the audio can start playing.

offCanplay

This method is called using InnerAudioContext.offCanplay(function listener).

Feature description: Removes the listener for the event when the audio can start playing.

Parameter and description:function listener. The listener passed to onCanplay. If not provided, all listeners will be removed.

Example:

```
const listener = function (res) { console.log(res) }

InnerAudioContext.onCanplay(listener)
```

```
InnerAudioContext.offCanplay(listener) // Must pass the same function object used i
```

onPlay

This method is called using InnerAudioContext.onPlay(function listener).

Feature description: Listens for audio play event.
Parameter and description:function listener. The listener for audio play event.
offPlay

This method is called using InnerAudioContext.offPlay(function listener).

Feature description: Removes the listener for audio play event.
Parameter and description:function listener. The listener passed to onPlay. If not provided, all listeners will be removed.
Example:

```
const listener = function (res) { console.log(res) }

InnerAudioContext.onPlay(listener)
InnerAudioContext.offPlay(listener) // Must pass the same function object used in o
```

onPause

This method is called using InnerAudioContext.onPause(function listener).

Feature description: Listens for audio pause event.
Parameter and description:function listener. The listener for the audio pause event.
offPause

This method is called using InnerAudioContext.offPause(function listener).

Feature description: Removes the listener for the audio pause event.
Parameter and description:function listener. The listener passed to onPause. If not provided, all listeners will be removed.

©2013-2025 Tencent Cloud International Pte. Ltd.

Example:

```
const listener = function (res) { console.log(res) }

InnerAudioContext.onPause(listener)
InnerAudioContext.offPause(listener) // Must pass the same function object used in
```

onStop

This method is called using InnerAudioContext.onStop(function listener).

Feature description: Listens for audio stop event.

Parameter and description:function listener. The listener for audio stop event.

offStop

This method is called using InnerAudioContext.offStop(function listener).

Feature description: Removes the listener for audio stop event.

Parameter and description:function listener. The listener passed to onStop. If not provided, all listeners will be removed.

Example:

```
const listener = function (res) { console.log(res) }

InnerAudioContext.onStop(listener)
InnerAudioContext.offStop(listener) // Must pass the same function object used in o
```

onEnded

This method is called using InnerAudioContext.onEnded(function listener).

Feature description: Listens for the event when the audio naturally ends.

Parameter and description:function listener. The listener for the event when the audio naturally ends.

offEnded

This method is called using InnerAudioContext.offEnded(function listener).

Feature description: Removes the listener for the event when the audio naturally ends.

©2013-2025 Tencent Cloud International Pte. Ltd.

Parameter and description:function listener. The listener passed to onEnded. If not provided, all listeners will be removed.

Example:

```
const listener = function (res) { console.log(res) }

InnerAudioContext.onEnded(listener)
InnerAudioContext.offEnded(listener) // Must pass the same function object used in
```

onTimeUpdate

This method is called using InnerAudioContext.onTimeUpdate(function listener).

Feature description: Listens for audio playback progress update event.

Parameter and description:function listener. The listener for audio playback progress update events.

offTimeUpdate

This method is called using InnerAudioContext.offTimeUpdate(function listener).

Feature description: Removes the listener for audio playback progress update event.

Parameter and description:function listener. The listener passed to onTimeUpdate. If not provided, all listeners will be removed.

Example:

```
const listener = function (res) { console.log(res) }

InnerAudioContext.onTimeUpdate(listener)
InnerAudioContext.offTimeUpdate(listener) // Must pass the same function object use
```

onError

This method is called using InnerAudioContext.onError(function listener).

Feature description: Listens for audio playback error event.

Parameter and description:function listener. The listener for audio playback error event.

Object res

Property	Type	Description

errMsg	string	
errCode	number	

errCode valid values

Value	Description
10001	System error.
10002	Network error.
10003	File error.
10004	Format error.
-1	Unknown error.

Tips：
When errCode=100001, if errMsg contains INNERCODE -11828, check whether Content-Length is missing in the response header.
When errCode=100001, if errMsg contains systemErrCode: 200333420, check whether the file encoding format is consistent with fileExtension.

offError

> This method is called using InnerAudioContext.offError(function listener).

Feature description: Removes the listener for audio playback error event.
Parameter and description: function listener. The listener passed to onError. If not provided, all listeners will be removed.
Example:

```
const listener = function (res) { console.log(res) }

InnerAudioContext.onError(listener)
InnerAudioContext.offError(listener) // Must pass the same function object used in
```

onWaiting

> This method is called using InnerAudioContext.onWaiting(function listener).

Feature description: Listens for audio loading event. It will be triggered when the audio needs to stop and load due to insufficient data.
Parameter and description:function listener. The listener for audio loading event.
offWaiting

This method is called using InnerAudioContext.offWaiting(function listener).

Feature description: Removes the listener for audio loading event.
Parameter and description:function listener. The listener passed to onWaiting. If not provided, all listeners will be removed.
Example:

```
const listener = function (res) { console.log(res) }

InnerAudioContext.onWaiting(listener)
InnerAudioContext.offWaiting(listener) // Must pass the same function object used i
```

onSeeking

This method is called using InnerAudioContext.onSeeking(function listener).

Feature description: Listens for audio seeking event.
Parameter and description:function listener. The listener for audio seeking event.
offSeeking

This method is called using InnerAudioContext.offSeeking(function listener).

Feature description: Removes the listener for audio seeking event.
Parameter and description:function listener. The listener passed to onSeeking. If not provided, all listeners will be removed.
Example:

```
const listener = function (res) { console.log(res) }

InnerAudioContext.onSeeking(listener)
InnerAudioContext.offSeeking(listener) // Must pass the same function object used i
```

©2013-2025 Tencent Cloud International Pte. Ltd.

onSeeked

This method is called using InnerAudioContext.onSeeked(function listener).

Feature description: Listens for the event when audio seeking is completed.
Parameter and description:function listener. The listener for the event when audio seeking is completed.
offSeeked

This method is called using InnerAudioContext.offSeeked(function listener).

Feature description: Removes the listener for the event when audio seeking is completed.
Parameter and description:function listener. The listener passed to onSeeked. If not provided, all listeners will be removed.
Example:

```
const listener = function (res) { console.log(res) }

InnerAudioContext.onSeeked(listener)
InnerAudioContext.offSeeked(listener) // Must pass the same function object used in
```

WXML

Last updated：2025-03-20 18:01:56

createSelectorQuery

The application of this API is through SelectorQuery wx.createSelectorQuery().

Function Description: Returns an instance of the SelectorQuery object. In custom components or pages containing custom components, replace with `this.createSelectorQuery()` .

Return Value: SelectorQuery.

Sample Code:

```
const query = wx.createSelectorQuery()
query.select('#the-id').boundingClientRect()
query.selectViewport().scrollOffset()
query.exec(function (res) {
  res[0].top // The upper boundary coordinate of the #the-id node.
  res[1].scrollTop // The vertical scroll position of the display area.
})
```

createIntersectionObserver

The usage of this API is IntersectionObserver wx.createIntersectionObserver(Object component, Object options).

Function Description: Creates and returns an instance of the IntersectionObserver object. In custom components or pages containing custom components, `this.createIntersectionObserver([options])` should be used as a substitute.

Parameters and Description: Object this

Object component, a custom component instance;

Object options, with the following legal values:

Attributes	Types	Default value	Required	Note

thresholds	Array.<number>	[0]	Not required	An array of numerical values, encompassing all thresholds.
initialRatio	number	0	Not required	The initial intersection ratio, if the detected intersection ratio at the time of invocation is not equivalent to this value and reaches the threshold, it will trigger a callback function of the listener once.
observeAll	boolean	false	Not required	Whether to observe multiple target nodes simultaneously (rather than a single one). If set to true, the targetSelector of observe will select multiple nodes (be aware: selecting too many nodes at once may impact rendering performance).

Return Value: IntersectionObserver.

IntersectionObserver

IntersectionObserver object, used to infer whether certain nodes can be seen by the user and what percentage can be seen by the user.

.relativeTo

> The method is utilized as follows: IntersectionObserver.relativeTo(string selector, Object margins)

Function Description: Utilize a selector to designate a node, serving as one of the reference areas.
Parameters and Description:
string selector, the selector;
Object margins, the boundaries used to expand (or contract) the layout area of the reference node.

Attributes	Types	Default value	Required	Note
left	number	-	Not required	The left boundary of the node's layout area.
right	number	-	Not required	The right boundary of the node's layout area.
top	number	-	Not required	The upper boundary of the node's layout area.

bottom	number	-	Not required	The lower boundary of the node's layout area.

.relativeToViewport

The method is utilized as IntersectionObserver.relativeToViewport(Object margins).

Function Description: Designates the page display area as one of the reference regions.

Parameters and Description: Object margins, utilized to expand (or contract) the boundaries of the reference node's layout area.

Attributes	Types	Default value	Required	Note
left	number	-	Not required	The left boundary of the node's layout area.
right	number	-	Not required	The right boundary of the node's layout area.
top	number	-	Not required	The upper boundary of the node's layout area.
bottom	number	-	Not required	The lower boundary of the node's layout area.

Example Code

In the following example code, if the target node (specified by the selector .target-class) enters 100px below the display area, the callback function will be triggered.

```
Page({
  onLoad() {
    wx.createIntersectionObserver().rela
  onLoad: function(){
    wx.createIntersectionObserver().relativeToViewport({bottom: 100}).observe('.tar
      res.intersectionRatio // the ratio of the intersection region to the layout r
      res.intersectionRect // intersection region
      res.intersectionRect.left // coordinates of the left boundary of the intersec
      res.intersectionRect.top // the upper boundary of the intersection region
      res.intersectionRect.width // width of the intersection region
      res.intersectionRect.height // height of the intersection region
    })
  }
})
```

.disconnect

The method is utilized as IntersectionObserver.disconnect().

Function Description: Ceases monitoring, the callback function will no longer be triggered.

.observe

The method is utilized as IntersectionObserver.observe(string targetSelector, function callback).

Function Description: Designates the target node and commences monitoring for changes in intersection status.
Parameters and Description:
string targetSelector, selector;
function callback, a callback function that listens for changes in the intersection state, with the following arguments Object res:
Parameters for the Callback Function Monitoring Intersection Status Changes: Object res.

Structural attributes	Types	Note
intersectionRatio	number	Intersection Ratio
intersectionRect	object	Boundary of the Intersection Area
boundingClientRect	object	Target Boundary
relativeRect	object	Boundary of the Reference Area
time	number	Timestamp during Intersection Detection

Structure of res.intersectionRect

Structural attributes	Types	Note
left	number	Left Boundary
right	number	Right Boundary
top	number	Upper Boundary
bottom	number	Lower Boundary

©2013-2025 Tencent Cloud International Pte. Ltd.

Structure of res.boundingClientRect

Structural attributes	Types	Note
left	number	Left Boundary
right	number	Right Boundary
top	number	Upper Boundary
bottom	number	Lower Boundary

Structure of res.relativeRect

Structural attributes	Types	Note
left	number	Left Boundary
right	number	Right Boundary
top	number	Upper Boundary
bottom	number	Lower Boundary

NodesRef

An object used to obtain information about a WXML node.

.fields

The method is utilized as NodesRef.fields(Object fields, function callback)

Function Description: Get information about the node. The fields to be fetched are specified in fields. The return value is the selectorQuery corresponding to nodesRef.

Parameters and Description:

Object fields, the legal parameters are as follows.

Attributes	Types	Default	Required	Note

		value		
id	boolean	false	Not required	Whether to return the node id
dataset	boolean	false	Not required	Whether to return the node dataset
rect	boolean	false	Not required	Whether to return the layout position of the node (left, right, top, bottom)
size	boolean	false	Not required	Whether to return the dimensions of the node (width, height)
scrollOffset	boolean	false	Not required	Whether to return the `scrollLeft scrollTop` of the node, the node must be a `scroll-view` or `viewport`
properties	Array.<string>	-	Not required	Specify a list of attribute names to return the current attribute values of the corresponding node (only the regular attribute values marked in the component documentation can be obtained, the attribute values of id, class, style, and event binding cannot be retrieved)
computedStyle	Array.<string>	-	Not required	Specify a list of style names to return the current values of the corresponding style names for the node
context	boolean	false	Not required	Whether to return the corresponding Context object of the node

Note:

The priority of computedStyle is higher than size. If width/height is specified in computedStyle and size: true is passed in at the same time, the width/height obtained from computedStyle will be returned preferentially.

function callback, callback function, argument Object res is node related information.

Return Value: The selectorQuery corresponding to nodesRef.

Sample Code

```
Page({
getFields () {
  wx.createSelectorQuery().select('#the-id').fields({
    dataset: true,
    size: true,
    scrollOffset: true,
    properties: ['scrollX', 'scrollY'],
```

```
      computedStyle: ['margin', 'backgroundColor'],
      context: true, }, function (res) { {
    }, function (res) {
      res.dataset // the node's dataset
      res.width // width of the node
      res.height // the height of the node
      res.scrollLeft // The horizontal scroll position of the node.
      res.scrollTop // The vertical scroll position of the node.
      res.scrollX // The current value of the node's scroll-x property.
      res.scrollY // The current value of the node's scroll-y property.
      // This returns the name of the style to be returned.
      res.margin
      res.backgroundColor
      res.context // The node's corresponding Context object.
      res.ref // The Ref object for the node.
    res.ref // Ref object for the node }).exec()
  }
})
```

.boundingClientRect

The method is used as follows:SelectorQuery NodesRef.boundingClientRect(function callback)

Function Description: Adds a query request for the layout position of the node. Relative to the display area, in pixels. Its function is similar to the `getBoundingClientRect` of the DOM.

Parameters and Description: function callback**.** This is a callback function. After executing the SelectorQuery.exec method, the node information will be returned in the callback.

Object res

Attributes	Types	Note
id	string	Node's ID
dataset	Object	Node's dataset
left	number	Node's left boundary coordinate
right	number	Node's right boundary coordinate
top	number	Node's upper boundary coordinate
bottom	number	Node's lower boundary coordinate
width	number	Node's width

height	number	Node's height

Return Value: SelectorQuery.

Sample Code:

```
Page({
  getRect() {
    wx.createSelectorQuery().select('#the-id').boundingClientRect(function (rect) {
      rect.id // Node's ID
      rect.dataset // Node's dataset
      rect.left // Node's left boundary coordinate
      rect.right // Node's right boundary coordinate
      rect.top // Node's upper boundary coordinate
      rect.bottom // Node's lower boundary coordinate
      rect.width // Node's width
      rect.height // Node's height
    }).exec()
  },
  getAllRects() {
    wx.createSelectorQuery().selectAll('.a-class').boundingClientRect(function (rec
      rects.forEach(function (rect) {
        rect.id // Node's ID
        rect.dataset // Node's dataset
        rect.left // Node's left boundary coordinate
        rect.right // Node's right boundary coordinate
        rect.top // Node's upper boundary coordinate
        rect.bottom // Node's lower boundary coordinate
        rect.width // Node's width
        rect.height // Node's height
      })
    }).exec()
  }
})
```

.scrollOffset

The method is utilized as SelectorQuery NodesRef.scrollOffset(function callback)

Function Description: Adds a query request for the node's scroll position. Measured in pixels. The node must be a `scroll-view` or a `viewport` .

Parameters and Description: function callback, a callback function, after executing the SelectorQuery.exec method, the node information will be returned in the callback.

Object res

Attributes	Types	Note
id	string	Node's ID
dataset	Object	Node's dataset
scrollLeft	number	Node's horizontal scroll position
scrollTop	number	Node's vertical scroll position

Return Value: SelectorQuery.

Sample Code:

```
Page({
  getScrollOffset() {
    wx.createSelectorQuery().selectViewport().scrollOffset(function (res) {
      res.id // Node's ID
      res.dataset // Node's dataset
      res.scrollLeft // Node's horizontal scroll position
      res.scrollTop // Node's vertical scroll position
    }).exec()
  }
})
```

.context

The method is utilized as follows: SelectorQuery NodesRef.context(function callback)

Function Description: Adds a query request for the Context object of the node. Currently supports the acquisition of MapContext.

Parameters and Description: function callback, a callback function that returns node information after executing the SelectorQuery.exec method.

Object res。

Attributes	Types	Note
context	Object	Context object corresponding to the node

Return Value: SelectorQuery.

Sample Code:

```
Page({
  getContext() {
    wx.createSelectorQuery().select('.the-video-class').context(function (res) {
      console.log(res.context) // Context object corresponding to the node. For ins
    }).exec()
  }
})
```

.node

The method is employed as follows: SelectorQuery NodesRef.node(NodesRef.nodeCallback callback)

Function Description: Get Node node instance. Currently supports ScrollViewContext and Canvas.

Parameters and Description: function callback, a callback function that returns node information upon the execution of the SelectorQuery.exec method**.**

Object res

Attributes	Types	Note
node	Object	Node instance corresponding to the node

Return Value: SelectorQuery.

Sample Code:

```
Page({
  getNode() {
    wx.createSelectorQuery().select('.canvas').node(function(res){
      console.log(res.node) // Canvas instance corresponding to the node.
    }).exec()
  }
})
```

SelectorQuery

Object for querying node information.

.exec

The method is utilized as follows: NodesRef SelectorQuery.exec(function callback)

Function Description: Executes all requests. The results of the requests are arranged in an array according to the order of the requests and returned in the first parameter of the callback.
Parameters and Description: function callback, a callback function.
Return Value: NodesRef.

.in

The method is utilized as follows: SelectorQuery SelectorQuery.in(Component component)

Function Description: Modifies the selection range of the selector to be within the custom component, component. (Initially, the selector only selects nodes within the page range and does not select any nodes within custom components).
Parameters and Description: Component component, a custom component instance**.**
Return Value: SelectorQuery.
Sample Code:

```
Component({
  queryMultipleNodes() {
    const query = wx.createSelectorQuery().in(this)
    query.select('#the-id').boundingClientRect(function (res) {
      res.top // The upper boundary coordinate of the #the-id node within this comp
    }).exec()
  }
})
```

.select

The method is utilized as follows: NodesRef SelectorQuery.select(string selector)

Function Description: Selects the first node that matches the selector, selector, on the current page. Returns an instance of a NodesRef object, which can be used to retrieve node information.
Parameters and Description: string selector, the selector**.**
Return Value: NodesRef
Selector Syntax: The selector is similar to a CSS selector, but only supports the following syntax.

ID Selector: #the-id.
Class Selector (multiple can be specified consecutively): .a-class.another-class.
Child Selector: .the-parent > .the-child.
Descendant Selector: .the-ancestor .the-descendant.
Cross-component Descendant Selector: .the-ancestor >>> .the-descendant.
Union of Multiple Selectors: #a-node, .some-other-nodes.

.selectAll

The method is utilized as follows: NodesRef SelectorQuery.selectAll(string selector)

Function Description: Selects all nodes matching the selector 'selector' within the current page.
Parameters and Description: string selector, the selector**.**
Return Value: NodesRef
Selector Syntax: The selector is similar to a CSS selector, but only supports the following syntax.
ID Selector: #the-id.
Class Selector (multiple can be specified consecutively): .a-class.another-class.
Child Selector: .the-parent > .the-child.
Descendant Selector: .the-ancestor .the-descendant.
Cross-component Descendant Selector: .the-ancestor >>> .the-descendant.
Union of Multiple Selectors: #a-node, .some-other-nodes.

.selectViewport

The method is utilized as follows: NodesRef SelectorQuery.selectViewport()

Function Description: Selects the display area. This can be used to obtain information such as the size of the display area and the scroll position.
Return Value: NodesRef

MediaQueryObserver

MediaQueryObserver object, utilized for monitoring changes in the media query status, such as whether the interface's length and width are within a specified range.

.disconnect

The method is utilized as follows: MediaQueryObserver.disconnect()

Function Description: Ceases monitoring. The callback function will no longer be triggered.

.observe

The method is utilized as follows: MediaQueryObserver.observe(Object descriptor, function callback)

Function Description: Initiates monitoring of media query changes.

Parameters and Descriptions:

Object descriptor, a descriptor for the media query.

Attributes	Types	Default value	Required	Note
minWidth	number	-	Supported	Minimum page width (in pixels)
maxWidth	number	-	Supported	Maximum page width (in pixels)
width	number	-	Supported	Page width (in pixels)
minHeight	number	-	Supported	Minimum page height (in pixels)
maxHeight	number	-	Supported	Maximum page height (in pixels)
height	number	-	Supported	Page height (in pixels)
orientation	string	-	Supported	Screen orientation (landscape or portrait)

function callback, a callback function that listens for changes in the state of the media query. The arguments Object res are as follows:

Attributes	Types	Note
matches	boolean	Whether the current status of the page satisfies the specified media query.

File

Last updated：2025-03-25 18:15:56

saveFile

This API is called using wx.saveFile(Object object).

Note:

This API is supported in mini programs but not in mini games.

Note:

The size limit for local file storage is 10 MB.

Feature description: This API saves a file to the local storage.

Note:

The saveFile moves the temporary file, so the tempFilePath provided will no longer be available after a successful call.

Parameter and description: Object object

Property	Type	Default value	Required	Description
tempFilePath	string	-	True	The temporary path of the file to be saved.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.success callback function parameter: Object res

Property	Type	Description
savedFilePath	string	The path of the file after saving.

Example:

```
wx.chooseImage({
success(res) {
  const tempFilePaths = res.tempFilePaths
```

```
    wx.saveFile({
      tempFilePath: tempFilePaths[0],
      success(res) {
        const savedFilePath = res.savedFilePath
      }
    })
  }
})
```

removeSavedFile

This API is called using wx.removeSavedFile(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Deletes local cache files.

Parameter and description: Object object

Property	Type	Default value	Required	Description
filePath	string	-	True	The path to the file that needs to be deleted.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
wx.getSavedFileList({
  success(res) {
    if (res.fileList.length > 0) {
      wx.removeSavedFile({
        filePath: res.fileList[0].filePath,
        complete(res) {
          console.log(res)
        }
      })
    }
```

```
  }
})
```

openDocument

This API is called using wx.openDocument(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Opens a document in a new page.

Parameter and description: Object object

Property	Type	Default value	Required	Description
filePath	string	-	True	The file path (local path), which can be obtained via downloadFile.
fileType	string	-	False	The type of the file, used to specify the file type for opening.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Valid values for object.fileType

Value	Description
pdf	PDF format.

Example:

```
wx.downloadFile({
  // Example URL, not a real file
  url: 'https://example.com/somefile.pdf',
  success(res) {
    const filePath = res.tempFilePath
```

```
    wx.openDocument({
      filePath,
      success(res) {
        console.log('Document opened successfully')
      }
    })
  }
})
```

getSavedFileList

This API is called using wx.getSavedFileList(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Gets the list of saved local cache files in this mini program.

Parameter and description: Object object

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.success callback function parameter: Object res

Property	Type	Description
fileList	Array.<Object>	Array of files, each item is a FileItem.

res.fileList structure

Property	Type	Description
filePath	string	Local path.
size	number	Local file size in bytes.

createTime	number	Timestamp when the file was saved, in seconds since 1970/01/01 08:00:00

Example:

```
wx.getSavedFileList({
  success(res) {
    console.log(res.fileList)
  }
})
```

getSavedFileInfo

This API is called using wx.getSavedFileInfo(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Gets the information of a local file. This API can only be used to get information about files that have been saved locally. For temporary files, use the wx.getFileInfo() .

Parameter and description: Object object

Property	Type	Default value	Required	Description
filePath	string	-	True	File path.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.success callback function parameter: Object res

Property	Type	Description
size	number	File size in bytes.
createTime	number	Timestamp when the file was saved, in seconds since 1970/01/01 08:00:00

©2013-2025 Tencent Cloud International Pte. Ltd.

Example:

```
wx.getSavedFileList({
  success(res) {
    console.log(res.fileList)
  }
})
```

getFileInfo

This API is called using wx.getFileInfo(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Gets the file information.

Parameter and description: Object object

Property	Type	Default value	Required	Description
filePath	string	-	True	Local file path.
digestAlgorithm	string	'md5'	False	The algorithm used to calculate the file digest.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Valid values for object.digestAlgorithm

Value	Description
md5	The MD5 algorithm.
sha1	The SHA1 algorithm.

object.success callback function parameter: Object res

Property	Type	Description

size	number	The file size in bytes.
digest	string	File digest calculated using the specified algorithm.

Example:

```
wx.getFileInfo({
  success(res) {
    console.log(res.size)
    console.log(res.digest)
  }
})
```

getFileSystemManager

This API is called using FileSystemManager wx.getFileSystemManager().

Feature description: Gets a globally unique file manager.

Return value: FileSystemManager , the file manager.

FileSystemManager

File manager, which can be obtained using wx.getFileSystemManager .

.access

This method is called using FileSystemManager.access(Object object).

Feature description: Determines whether a file or directory exists.

Parameter and description: Object object

Property	Type	Default value	Required	Description
path	string	-	True	The path of the file or directory to check for existence.
success	function	-	False	Callback function for successful API calls.

fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.fail callback function parameter: Object res

Property	Type	Description
errMsg	string	Error message.

Valid values for res.errMsg

Value	Description
fail no such file or directory ${path}	The file or directory does not exist.
bad file descriptor	Invalid file descriptor.
permission denied	Permission error. The file is read-only or write-only.
permission denied, cannot access file path	No access to the target path (e.g., usr directory)
not a directory	The specified path is not a directory
Invalid argument	Invalid argument, check if length or offset is out of range.
directory not empty	The directory is not empty.
the maximum size of the file storage limit is exceeded	Storage space is insufficient, or file size exceeds the limit (100 MB).
base64 encode error	Character encoding conversion failed (e.g., incorrect base64 format).
data to write is empty	The data to write is empty.
illegal operation on a directory	Illegal operation on a directory (e.g., specified filePath is an existing directory).
file already exists ${dirPath}	A file or directory with the same name already exists.
value of length is out of range	The length value is out of range.
value of offset is out of range	The offset value is out of range.
value of position is out of range	The position value is out of range.

Example:

```
const fs = wx.getFileSystemManager()
// Determine whether a file or directory exists
fs.access({
  path: `${wx.env.USER_DATA_PATH}/hello.txt`,
  success(res) {
    // The file exists
    console.log(res)
  },
  fail(res) {
    // File does not exist or other error
    console.error(res)
  }
})

// Synchronous API
try {
  fs.accessSync(`${wx.env.USER_DATA_PATH}/hello.txt`)
} catch(e) {
  console.error(e)
}
```

.accessSync

This method is called using FileSystemManager.accessSync(string path).

Feature description: Synchronously performs the same feature as FileSystemManager.access.

Parameter and description: string path The path of the file or directory to check for existence.

Example:

```
const fs = wx.getFileSystemManager()
// Determine whether a file or directory exists
fs.access({
  path: `${wx.env.USER_DATA_PATH}/hello.txt`,
  success(res) {
    // The file exists
    console.log(res)
  },
  fail(res) {
    // File does not exist or other error
    console.error(res)
  }
```

```
})

// Synchronous API
try {
  fs.accessSync(`${wx.env.USER_DATA_PATH}/hello.txt`)
} catch(e) {
  console.error(e)
}
```

.appendFile

This method is called using FileSystemManager.appendFile(Object object).

Feature description: Appends content to the end of a file.

Parameter and description: Object object

Property	Type	Default value	Required	Description
filePath	string	-	True	Path of the file to append content to.
data	string/ArrayBuffer	-	True	Text or binary data to append.
encoding	string	utf-8	False	Character encoding for writing to the file.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Valid values for object.encoding

Value	Description
ascii	-
base64	-
binary	-
hex	-
ucs2/ucs-2/utf16le/utf-16le	Reads in little-endian order.

utf-8/utf8	-
latin1	-

object.fail callback function parameter: Object res

Property	Type	Description
errMsg	string	Error message.

Valid values for res.errMsg

Value	Description
fail no such file or directory, open ${filePath}	The specified filePath does not exist.
fail illegal operation on a directory, open "${filePath}"	The specified filePath is a directory that already exists.
fail permission denied, open ${dirPath}	No write permission for the specified filePath.
fail sdcard not mounted	Failed to mount the Android SDCard.

Example:

```
const fs = wx.getFileSystemManager()

fs.appendFile({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  data: 'some text',
  encoding: 'utf8',
  success(res) {
    console.log(res)
  },
  fail(res) {
    console.error(res)
  }
})

// Synchronous API
try {
  fs.appendFileSync(`${wx.env.USER_DATA_PATH}/hello.txt`, 'some text', 'utf8')
} catch(e) {
  console.error(e)
}
```

.appendFileSync

This method is called using FileSystemManager.appendFileSync(string filePath, string|ArrayBuffer data, string encoding).

Feature description: Synchronously performs the same feature as FileSystemManager.appendFile.

Parameter and description

string filePath: Path of the file to append content to.

string|ArrayBuffer data: Text or binary data to append.

string encoding: Character encoding for writing to the file.

Valid values for encoding

Value	Description
ascii	-
base64	-
binary	-
hex	-
ucs2/ucs-2/utf16le/utf-16le	Reads in little-endian order.
utf-8/utf8	-
latin1	-

Example:

```
const fs = wx.getFileSystemManager()

fs.appendFile({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  data: 'some text',
  encoding: 'utf8',
  success(res) {
    console.log(res)
  },
  fail(res) {
    console.error(res)
  }
})
```

```
// Synchronous API
try {
  fs.appendFileSync(`${wx.env.USER_DATA_PATH}/hello.txt`, 'some text', 'utf8')
} catch(e) {
  console.error(e)
}
```

.close

This method is called using FileSystemManager.close(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Closes a file.

Parameter and description: Object object

Property	Type	Default value	Required	Description
fd	string	-	True	File descriptor to be closed, obtained through FileSystemManager.open or FileSystemManager.openSync.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example

```
const fs = wx.getFileSystemManager()
// Open a file
fs.open({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  flag: 'a+',
  success(res) {
    // Close the file
    fs.close({
      fd: res.fd
    })
  }
})
```

.closeSync

This method is called using FileSystemManager.closeSync(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Synchronously closes a file.

Parameter and description: Object object

Property	Type	Default value	Required	Description
fd	string	-	True	File descriptor to be closed, obtained through FileSystemManager.open or FileSystemManager.openSync.

Return value:undefined.

Example

```
const fs = wx.getFileSystemManager()
const fd = fs.openSync({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  flag: 'a+'
})

// Close the file
fs.closeSync({fd: fd})
```

.copyFile

This method is called using FileSystemManager.copyFile(Object object).

Feature description: Copies a file.

Parameter and description: Object object

Property	Type	Default value	Required	Description
srcPath	string	-	True	Source file path, must be a regular file.

destPath	string	-	True	Destination file path, supports local paths.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.fail callback function parameter: Object res

Property	Type	Description
errMsg	string	Error message.

Valid values for res.errMsg

Value	Description
fail permission denied, copyFile ${srcPath} -> ${destPath}	No write permission for the destination file path.
fail no such file or directory, copyFile ${srcPath} -> ${destPath}	Source file does not exist, or the parent directory of the destination file path does not exist.

Example:

```
const fs = wx.getFileSystemManager()
fs.copyFile({
  srcPath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  destPath: `${wx.env.USER_DATA_PATH}/hello_copy.txt`
  success(res) {
    console.log(res)
  },
  fail(res) {
    console.error(res)
  }
})

// Synchronous API
try {
  fs.copyFileSync(
    `${wx.env.USER_DATA_PATH}/hello.txt`,
    `${wx.env.USER_DATA_PATH}/hello_copy.txt`
  )
} catch(e) {
  console.error(e)
```

```
}
```

.copyFileSync

This method is called using FileSystemManager.copyFileSync(string srcPath, string destPath).

Feature description: Synchronously performs the same feature as FileSystemManager.copyFile.

Parameter and description:

string srcPath: Source file path, must be a regular file.

string destPath: Destination file path.

Example:

```
const fs = wx.getFileSystemManager()
fs.copyFile({
  srcPath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  destPath: `${wx.env.USER_DATA_PATH}/hello_copy.txt`
  success(res) {
    console.log(res)
  },
  fail(res) {
    console.error(res)
  }
})

// Synchronous API
try {
  fs.copyFileSync(
    `${wx.env.USER_DATA_PATH}/hello.txt`,
    `${wx.env.USER_DATA_PATH}/hello_copy.txt`
  )
} catch(e) {
  console.error(e)
}
```

.getFileInfo

This method is called using FileSystemManager.getFileInfo(Object object).

Feature description: Gets the information of local temporary file or local cache file in this mini program.

©2013-2025 Tencent Cloud International Pte. Ltd.

Parameter and description: Object object

Property	Type	Default value	Required	Description
filePath	string	-	True	The path of the file to read.
digestAlgorithm	string	md5	False	The algorithm used to calculate the file digest.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Valid values for object.digestAlgorithm

Value	Description
md5	The MD5 algorithm.
sha1	The SHA1 algorithm.

object.success callback function parameter: Object res

Property	Type	Description
size	number	The file size in bytes.
digest	string	File digest calculated using the specified algorithm.

object.fail callback function parameter: Object res

Property	Type	Description
errMsg	string	Error message.

Valid values for res.errMsg

Value	Description
fail file not exist	The specified filePath could not be found.
no such file or directory ${path}	File/directory does not exist, or the parent directory is missing.
Input/output error	Input/output stream is unavailable.

permission denied	Permission error. The file is read-only or write-only.
Path permission denied	No permission for the specified path.
not a directory	The specified path is not a directory.
Invalid argument	Invalid argument, check if length or offset is out of range.
excced max concurrent fd limit	Maximum number of file descriptors reached.

.fstat

This method is called using FileSystemManager.fstat(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Gets status information of a file.

Parameter and description: Object object

Property	Type	Default value	Required	Description
fd	string	-	True	File descriptor obtained through FileSystemManager.open or FileSystemManager.openSync.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.success callback function parameter: Object res

Property	Type	Description
stats	Stats	Stats object containing file status information.

Example:

```
const fs = wx.getFileSystemManager()
// Open a file
fs.open({
```

```
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  flag: 'a+',
  success(res) {
    // Get status information of the file
    fs.fstat({
      fd: res.fd,
      success(res) {
        console.log(res.stats)
      }
    })
  }
})
```

.fstatSync

This method is called using Stats FileSystemManager.fstatSync(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Synchronously gets status information of a file.

Parameter and description: Object object

Property	Type	Default value	Required	Description
fd	string	-	True	File descriptor obtained through FileSystemManager.open or FileSystemManager.openSync.

Return value: Stats Stats object containing file status information.

Example:

```
const fs = wx.getFileSystemManager()
const fd = fs.openSync({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  flag: 'a+'
})
const stats = fs.fstatSync({fd: fd})
console.log(stats)
```

.ftruncate

This method is called using FileSystemManager.ftruncate(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Truncates the file content.

Parameter and description: Object object

Property	Type	Default value	Required	Description
fd	string	-	True	File descriptor obtained through FileSystemManager.open or FileSystemManager.openSync.
length	number	-	True	The truncation position. Default value: 0. If the length is less than the file length (in bytes), only the first length bytes will be remained in the file, and the rest will be deleted. If the length is greater than the file length, the file will be extended, and the extended part will be filled with null bytes ('\\0').
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
const fs = wx.getFileSystemManager()
// Open a file
fs.open({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  flag: 'a+',
  success(res) {
    // Truncate the file content
    fs.ftruncate({
      fd: res.fd,
      length: 10, // Truncate the file from the 10th byte
      success(res) {
        console.log(res)
      }
    })
  }
})
```

.ftruncateSync

This method is called using undefined FileSystemManager.ftruncateSync(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Synchronously truncates the file content.

Parameter and description: Object object

Property	Type	Default value	Required	Description
fd	string	-	True	File descriptor obtained through FileSystemManager.open or FileSystemManager.openSync.
length	number	-	True	The truncation position. Default value: 0. If the length is less than the file length (in bytes), only the first length bytes will be remained in the file, and the rest will be deleted. If the length is greater than the file length, the file will be extended, and the extended part will be filled with null bytes ('\\0').

Return value: undefined.

Example:

```
const fs = wx.getFileSystemManager()
const fd = fs.openSync({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  flag: 'a+'
})
fs.ftruncateSync({
  fd: fd,
  length: 10 // Truncate the file from the 10th byte
})
```

.getSavedFileList

This method is called using FileSystemManager.getSavedFileList(Object object).

Feature description: Gets the list of saved local cache files in this mini program.

Parameter and description: Object object

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.success callback function parameter: Object res

Property	Type	Description
fileList	Array.<Object>	Array of file objects.

res.fileList structure

Property	Type	Description
filePath	string	Local path.
size	number	Local file size in bytes.
createTime	number	Timestamp when the file was saved, in seconds since 1970/01/01 08:00:00

.mkdir

This method is called using FileSystemManager.mkdir(Object object).

Feature description: Creates a directory.

Parameter and description: Object object

Property	Type	Default value	Required	Description
dirPath	string	-	True	Path of the directory to create.
recursive	boolean	false	False	Whether to recursively create the parent directories. If the parent directories already exist, they will not be created. For example, if dirPath is a/b/c/d and recursive is true, it

				will create directory a, then b inside a, and so on until d inside a/b/c.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.fail callback function parameter: Object res

Property	Type	Description
errMsg	string	Error message.

Valid values for res.errMsg

Value	Description
fail no such file or directory ${dirPath}	The parent directory does not exist.
fail permission denied, open ${dirPath}	No write permission for the specified filePath.
fail file already exists ${dirPath}	A file or directory with the same name already exists.

Example:

```
const fs = wx.getFileSystemManager()
fs.mkdir({
  dirPath: `${wx.env.USER_DATA_PATH}/example`,
  recursive: false,
  success(res) {
    console.log(res)
  },
  fail(res) {
    console.error(res)
  }
})

// Synchronous API
try {
  fs.mkdirSync(`${wx.env.USER_DATA_PATH}/example`, false)
} catch(e) {
  console.error(e)
}
```

.mkdirSync

This method is called using FileSystemManager.mkdirSync(string dirPath, boolean recursive).

Feature description: Synchronously performs the same feature as FileSystemManager.mkdir.

Parameter and description:

string dirPath: Path of the directory to create.

boolean recursive: Whether to recursively create all the directory's parent directories before creating the directory. If the parent directories already exist, they will not be created. For example, if dirPath is a/b/c/d and recursive is true, it will create directory a, then b inside a, and so on until d inside a/b/c.

Example:

```
const fs = wx.getFileSystemManager()
fs.mkdir({
  dirPath: `${wx.env.USER_DATA_PATH}/example`,
  recursive: false,
  success(res) {
    console.log(res)
  },
  fail(res) {
    console.error(res)
  }
})

// Synchronous API
try {
  fs.mkdirSync(`${wx.env.USER_DATA_PATH}/example`, false)
} catch(e) {
  console.error(e)
}
```

.open

This method is called using FileSystemManager.open(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Opens a file and returns a file descriptor.

Parameter and description: Object object

Property	Type	Default value	Required	Description
filePath	string	-	True	File path (local path).
flag	string	r	False	File system flag. Default value: r.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Valid values for flag

Value	Description
a	Opens the file for appending. If the file does not exist, it will be created.
ax	Similar to 'a', but the file opening will fail if the path exists.
a+	Opens the file for reading and appending. If the file does not exist, it will be created.
ax+	Similar to 'a+', but the file opening will fail if the path exists.
as	Open file for appending in synchronous mode. If the file does not exist, it will be created.
as+	Open file for reading and appending in synchronous mode. If the file does not exist, it will be created.
r	Opens the file for reading. If the file does not exist, an error will occur.
r+	Opens the file for reading and writing. If the file does not exist, an error will occur.
w	Opens the file for writing. If the file does not exist, create the file; if the file exists, truncate the file.
wx	Similar to 'w', but the file opening will fail if the path exists.
w+	Opens the file for reading and writing. If the file does not exist, create the file; if the file exists, truncate the file.
wx+	Similar to 'w+', but the file opening will fail if the path exists.

object.success callback function parameter: Object res

Property	Type	Description

fd	string	File descriptor.

Example:

```
const fs = wx.getFileSystemManager()
fs.open({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  flag: 'a+',
  success(res) {
    console.log(res.fd)
  }
})
```

.openSync

This method is called using string FileSystemManager.openSync(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Synchronously opens the file and returns the file descriptor.

Parameter and description: Object object

Property	Type	Default value	Required	Description
filePath	string	-	True	File path (local path).
flag	string	r	False	File system flag. Default value: r.

Valid values for flag

Value	Description
a	Opens the file for appending. If the file does not exist, it will be created.
ax	Similar to 'a', but the file opening will fail if the path exists.
a+	Opens the file for reading and appending. If the file does not exist, it will be created.
ax+	Similar to 'a+', but the file opening will fail if the path exists.
as	Open file for appending in synchronous mode. If the file does not exist, it will be created.

©2013-2025 Tencent Cloud International Pte. Ltd.

as+	Open file for reading and appending in synchronous mode. If the file does not exist, it will be created.
r	Opens the file for reading. If the file does not exist, an error will occur.
r+	Opens the file for reading and writing. If the file does not exist, an error will occur.
w	Opens the file for writing. If the file does not exist, create the file; if the file exists, truncate the file.
wx	Similar to 'w', but the file opening will fail if the path exists.
w+	Opens the file for reading and writing. If the file does not exist, create the file; if the file exists, truncate the file.
wx+	Similar to 'w+', but the file opening will fail if the path exists.

Return value:string, a file descriptor.

Example:

```
const fs = wx.getFileSystemManager() const fd = fs.openSync({ filePath: `${wx.env.U
```

.read

This method is called using FileSystemManager.read(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Reads a file.

Parameter and description: Object object

Property	Type	Default value	Required	Description
fd	string	-	True	File descriptor obtained through FileSystemManager.open or FileSystemManager.openSync.
arrayBuffer	ArrayBuffer	-	True	The buffer to which the data is written, and it must be an instance of ArrayBuffer.
offset	number	0	False	The write offset in the buffer. Default value: 0.
length	number	0	False	The number of bytes to be read in the file. Default value:

©2013-2025 Tencent Cloud International Pte. Ltd.

				0.
position	number	-	False	The starting position for reading the file. If this property is not provided or null is entered, the file is read from the position of the current file pointer. If the position is a positive integer, the file pointer position remains unchanged and the file is read from the position.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.success callback function parameter: Object res

Property	Type	Description
bytesRead	number	Number of bytes actually read.
arrayBuffer	ArrayBuffer	The buffer object that was written to, same as the arrayBuffer parameter passed to the API.

Example:

```
const fs = wx.getFileSystemManager()
const ab = new ArrayBuffer(1024)
// Open a file
fs.open({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  flag: 'a+',
  success(res) {
    // Read the file into an ArrayBuffer
    fs.read({
      fd: res.fd,
      arrayBuffer: ab,
      length: 10,
      success(res) {
        console.log(res)
      }
    })
  }
})
```

.readSync

This method is called using ReadResult FileSystemManager.readSync(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Reads a file.

Parameter and description: Object object

Property	Type	Default value	Required	Description
fd	string	-	True	File descriptor obtained through FileSystemManager.open or FileSystemManager.openSync.
arrayBuffer	ArrayBuffer	-	True	The buffer to which the data is written, and it must be an instance of ArrayBuffer.
offset	number	0	False	The write offset in the buffer. Default value: 0.
length	number	0	False	The number of bytes to be read in the file. Default value: 0.
position	number	-	False	The starting position for reading the file. If this property is not provided or null is entered, the file is read from the position of the current file pointer. If the position is a positive integer, the file pointer position remains unchanged and the file is read from the position.

Return value: ReadResult. The result of reading the files.

Example:

```
const fs = wx.getFileSystemManager()
const ab = new ArrayBuffer(1024)
const fd = fs.openSync({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  flag: 'a+'
})
const res = fs.readSync({
  fd: fd,
  arrayBuffer: ab,
  length: 10
})
console.log(res)
```

.readCompressedFile

This method is called using FileSystemManager.readCompressedFile(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Reads the content of a local file with a specified compression type.

Parameter and description: Object object

Property	Type	Default value	Required	Description
filePath	string	-	True	Path of the file to read (local user file or code package file).
compressionAlgorithm	string	-	True	File compression type, currently only supports br.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Valid values for object.compressionAlgorithm:

Value	Description
br	Brotli-compressed file.

object.success callback function parameter: Object res

Property	Type	Description
data	ArrayBuffer	File content.

Example:

```
const fs = wx.getFileSystemManager()

// Asynchronous API
fs.readCompressedFile({
```

```
  filePath: '${wx.env.USER_DATA_PATH}/hello.br',
  compressionAlgorithm: 'br',
  success(res) {
    console.log(res.data)
  },
  fail(res) {
    console.log('readCompressedFile fail', res)
  }
})

// Synchronous API
const data = fs.readCompressedFileSync({
  filePath: '${wx.env.USER_DATA_PATH}/hello.br',
  compressionAlgorithm: 'br',
})
console.log(data)
```

.readCompressedFileSync

This method is called using ArrayBuffer FileSystemManager.readCompressedFileSync(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Synchronously reads the local file contents of specified compression type.

Parameter and description: Object object

Property	Type	Default value	Required	Description
filePath	string	-	True	Path of the file to read (local user file or code package file).
compressionAlgorithm	string	-	True	File compression type, currently only supports br.

Valid values for object.compressionAlgorithm:

Value	Description
br	Brotli-compressed file.

Return value: ArrayBuffer, the result of reading the files.

Example:

```
const fs = wx.getFileSystemManager()

// Asynchronous API
fs.readCompressedFile({
  filePath: '${wx.env.USER_DATA_PATH}/hello.br',
  compressionAlgorithm: 'br',
  success(res) {
    console.log(res.data)
  },
  fail(res) {
    console.log('readCompressedFile fail', res)
  }
})

// Synchronous API
try {
  const data = fs.readCompressedFileSync({
    filePath: '${wx.env.USER_DATA_PATH}/hello.br',
    compressionAlgorithm: 'br',
  })
  console.log(data)
} catch (err) {
  console.log(err)
}
```

.readdir

This method is called using FileSystemManager.readdir(Object object).

Feature description: Reads the list of files in the directory.

Parameter and description: Object object

Property	Type	Default value	Required	Description
dirPath	string	-	True	The path to the directory to be read.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends

				(regardless of success or failure).

object.success callback function parameter: Object res

Property	Type	Description
files	Array.<string>	An array of filenames in a specified directory.

object.fail callback function parameter: Object res

Property	Type	Description
errMsg	string	Error message.

Valid values for res.errMsg

Value	Description
fail no such file or directory ${dirPath}	The directory does not exist.
fail not a directory ${dirPath}	The dirPath is not a directory.
fail permission denied, open ${dirPath}	No read permission for the specified filePath.

Example:

Note:

The readdir API cannot access the root path of the file system (wxfile://).

```
const fs = wx.getFileSystemManager()
fs.readdir({
  dirPath: `${wx.env.USER_DATA_PATH}/example`,
  success(res) {
    console.log(res.files)
  },
  fail(res) {
    console.error(res)
  }
})

// Synchronous API
try {
  const res = fs.readdirSync(`${wx.env.USER_DATA_PATH}/example`)
  console.log(res)
} catch(e) {
  console.error(e)
```

```
}
```

.readdirSync

This method is called using Array.<string>FileSystemManager.readdirSync(string dirPath).

Feature description: Synchronously performs the same feature as FileSystemManager.readdir.

Parameter and description:string dirPath, the path to the directory to be read.

Return value: Array.<string> files, an array of file names in a specified directory.

Example:

```
const fs = wx.getFileSystemManager()
fs.readdir({
  dirPath: `${wx.env.USER_DATA_PATH}/example`,
  success(res) {
    console.log(res.files)
  },
  fail(res) {
    console.error(res)
  }
})

// Synchronous API
try {
  const res = fs.readdirSync(`${wx.env.USER_DATA_PATH}/example`)
  console.log(res)
} catch(e) {
  console.error(e)
}
```

.readFile

This method is called using FileSystemManager.readFile(Object object).

Feature description: Reads the content of a local file. The maximum size of a single file is 100 MB.

Parameter and description: Object object

Property	Type	Default value	Required	Description

filePath	string	-	True	The path to the file to be read.
encoding	string	-	False	Specify the file's character encoding for reading. If no encoding is provided, the file's binary content will be read as an ArrayBuffer.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Valid values for object.encoding

Value	Description
ascii	-
base64	-
binary	-
hex	-
ucs2/ucs-2/utf16le/utf-16le	Reads in little-endian order.
utf-8/utf8	-
latin1	-

object.success callback function parameter: Object res

Property	Type	Description
data	string/ArrayBuffer	File content.

object.fail callback function parameter: Object res

Property	Type	Description
errMsg	string	Error message.

Valid values for res.errMsg

Value	Description

fail no such file or directory, open ${filePath}	The specified filePath directory does not exist.
fail permission denied, open ${dirPath}	No read permission for the specified filePath.

Example:

```
const fs = wx.getFileSystemManager()
fs.readFile({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  encoding: 'utf8',
  position: 0,
  success(res) {
    console.log(res.data)
  },
  fail(res) {
    console.error(res)
  }
})

// Synchronous API
try {
  const res = fs.readFileSync(`${wx.env.USER_DATA_PATH}/hello.txt`, 'utf8', 0)
  console.log(res)
} catch(e) {
  console.error(e)
}
```

.readFileSync

This method is called using string|ArrayBuffer FileSystemManager.readFileSync(string filePath, string encoding).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Synchronously performs the same feature as FileSystemManager.readFile.

Parameter and description:

string filePath: The path to the file to be read.

string encoding: Specifies the file's character encoding for reading. If no encoding is provided, the file's binary content will be read as an ArrayBuffer.

Valid values for encoding

Value	Description

ascii	-
base64	-
binary	-
hex	-
ucs2/ucs-2/utf16le/utf-16le	Reads in little-endian order.
utf-8/utf8	-
latin1	-

Return value:string|ArrayBuffer data, file content.

Example:

```
const fs = wx.getFileSystemManager()
fs.readFile({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  encoding: 'utf8',
  position: 0,
  success(res) {
    console.log(res.data)
  },
  fail(res) {
    console.error(res)
  }
})

// Synchronous API
try {
  const res = fs.readFileSync(`${wx.env.USER_DATA_PATH}/hello.txt`, 'utf8', 0)
  console.log(res)
} catch(e) {
  console.error(e)
}
```

.readZipEntry

This method is called using FileSystemManager.readZipEntry(Object object).

Feature description: Read files in a compressed package.

Parameter and description: Object object

Property	Type	Default value	Required	Description
filePath	string	-	True	The path of the zip file to be read (local path).
encoding	string	-	False	Specifies the character encoding for reading files. This is only effective when the entries value is "all". If entries is "all" and no encoding is provided, the file's binary content will be read as an ArrayBuffer.
entries	Array.<Object>/'all'	-	True	The list of files within the zip archive to be read (passing "all" means reading all files in the archive).
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Valid values for encoding

Value	Description
ascii	-
base64	-
binary	-
hex	-
ucs2/ucs-2/utf16le/utf-16le	Reads in little-endian order.
utf-8/utf8	-
latin1	-

entries structural property

Structural property	Type	Default value	Required	Description
path	string	-	True	The path to the files in the compressed package.

encoding	string	-	False	Specifies the file's character encoding for reading. If no encoding is provided, the file's binary content will be read as an ArrayBuffer. The valid values are the same as those for encoding.
position	number	-	False	The position to start reading from within the file. If not specified, reading starts from the beginning. The range should be [position, position+length). Valid range: [0, fileLength - 1]. Unit: byte.
length	number	-	False	The length of the file to read. If not specified, reads until the end of the file. Valid range: [1, fileLength]. Unit: byte.

object.success callback function parameter: Object res

Property	Type	Description
entries	Object	The result of reading the files. res.entries is an object where the key is the file path and the value is a FileItem object representing the read result. Each FileItem contains data (file content) and errMsg (error message) properties.

entries structural property

Structural property	Type	Description
path	string	File path.

path structural property

Structural property	Type	Description
data	string/ArrayBuffer	File content.
errMsg	string	Error message.

Example:

```
const fs = wx.getFileSystemManager()
// Read one or more files in the zip package
fs.readZipEntry({
  filePath: 'wxfile://from/to.zip',
  entries: [{
    path: 'some_folder/my_file.txt', // File path within the zip package
```

```
    encoding: 'utf-8', // Specify the file's character encoding for reading. If no
    position: 0, // The position to start reading from within the file. If not spec
    length: 10000, // The length of the file to read. If not specified, reads until
  }, {
    path: 'other_folder/orther_file.txt', // The path to the files within the zip p
  }],
  success(res) {
    console.log(res.entries)
    // res.entries === {
    // 'some_folder/my_file.txt': {
    // errMsg: 'readZipEntry:ok',
    // data: 'xxxxxx'
    // },
    // 'other_folder/orther_file.txt': {
    // data: (ArrayBuffer)
    // }
    // }
  },
  fail(res) {
    console.log(res.errMsg)
  },
})

// Read all files in the zip package Allow specifying a unified encoding. position
fs.readZipEntry({
  filePath: 'wxfile://from/to.zip',
  entries: 'all'
  encoding: 'utf-8', // Specify the file's character encoding for reading. If no en
  success(res) {
    console.log(res.entries)
    // res.entries === {
    // 'some_folder/my_file.txt': {
    // errMsg: 'readZipEntry:ok',
    // data: 'xxxxxx'
    // },
    // 'other_folder/orther_file.txt': {
    // errMsg: 'readZipEntry:ok',
    // data: 'xxxxxx'
    // }
    // }
  },
  fail(res) {
    console.log(res.errMsg)
  },
})
```

.removeSavedFile

This method is called using FileSystemManager.removeSavedFile(Object object).

Feature description: Deletes saved local cache files in this mini program.

Parameter and description: Object object

Property	Type	Default value	Required	Description
filePath	string	-	True	The path to the file that needs to be deleted.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.fail callback function parameter: Object res

Property	Type	Description
errMsg	string	Error message.

Valid values for res.errMsg

Value	Description
fail file not exist	The file cannot be found through the specified tempFilePath.

.rename

This method is called using FileSystemManager.rename(Object object).

Feature description: Renames a file. The files can be moved from oldPath to newPath.

Parameter and description: Object object

Property	Type	Default value	Required	Description
oldPath	string	-	True	The source file path, which can be a regular file or

				directory, supporting local paths.
newPath	string	-	True	The new file path, supporting local paths.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.fail callback function parameter: Object res

Property	Type	Description
errMsg	string	Error message.

Valid values for res.errMsg

Value	Description
fail permission denied, rename ${oldPath} -> ${newPath}	The specified source or target file does not have write permissions.
fail no such file or directory, rename ${oldPath} -> ${newPath}	Source file does not exist, or the parent directory of the destination file path does not exist.

Example:

```
const fs = wx.getFileSystemManager()
fs.rename({
  oldPath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  newPath: `${wx.env.USER_DATA_PATH}/hello_new.txt`,
  success(res) {
    console.log(res)
  },
  fail(res) {
    console.error(res)
  }
})

// Synchronous API
try {
  const res = fs.renameSync(
    `${wx.env.USER_DATA_PATH}/hello.txt`,
    `${wx.env.USER_DATA_PATH}/hello_new.txt`
```

```
  )
  console.log(res)
} catch(e) {
  console.error(e)
}
```

.renameSync

This method is called using FileSystemManager.renameSync(string oldPath, string newPath).

Feature description: Synchronously performs the same feature as FileSystemManager.rename.

Parameter and description

string oldPath: The source file path, which can be a regular file or directory, supporting local paths.

string newPath: The new file path, supporting local paths.

Example:

```
const fs = wx.getFileSystemManager()
fs.rename({
  oldPath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  newPath: `${wx.env.USER_DATA_PATH}/hello_new.txt`,
  success(res) {
    console.log(res)
  },
  fail(res) {
    console.error(res)
  }
})

// Synchronous API
try {
  const res = fs.renameSync(
    `${wx.env.USER_DATA_PATH}/hello.txt`,
    `${wx.env.USER_DATA_PATH}/hello_new.txt`
  )
  console.log(res)
} catch(e) {
  console.error(e)
}
```

.rmdir

This method is called using FileSystemManager.rmdir(Object object).

Feature description: Deletes a directory.

Parameter and description: Object object

Property	Type	Default value	Required	Description
dirPath	string	-	True	The path of the directory to be deleted.
recursive	boolean	false	False	Whether to recursively delete the directory. If true, deletes the directory and all its subdirectories and files.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.fail callback function parameter: Object res

Property	Type	Description
errMsg	string	Error message.

Valid values for res.errMsg

Value	Description
fail no such file or directory ${dirPath}	The directory does not exist.
fail directory not empty	The directory is not empty.
fail permission denied, open ${dirPath}	The specified dirPath does not have write permissions.

Example:

```
const fs = wx.getFileSystemManager()
fs.rmdir({
  dirPath: `${wx.env.USER_DATA_PATH}/example`,
  recursive: false,
  success(res) {
```

```
    console.log(res)
  },
  fail(res) {
    console.error(res)
  }
})

// Synchronous API
try {
  const res = fs.rmdirSync(`${wx.env.USER_DATA_PATH}/example`, false)
  console.log(res)
} catch(e) {
  console.error(e)
}
```

.rmdirSync

This method is called using FileSystemManager.rmdirSync(string dirPath, boolean recursive).

Feature description: Synchronously performs the same feature as FileSystemManager.rmdir.

Parameter and description:

string dirPath: The path of the directory to be deleted.

boolean recursive: Whether to recursively delete the directory. If true, deletes the directory and all its subdirectories and files.

Example:

```
const fs = wx.getFileSystemManager()
fs.rmdir({
  dirPath: `${wx.env.USER_DATA_PATH}/example`,
  recursive: false,
  success(res) {
    console.log(res)
  },
  fail(res) {
    console.error(res)
  }
})

// Synchronous API
try {
  const res = fs.rmdirSync(`${wx.env.USER_DATA_PATH}/example`, false)
  console.log(res)
} catch(e) {
```

```
  console.error(e)
}
```

.saveFile

This method is called using FileSystemManager.saveFile(Object object).

Feature description: Saves a temporary file to a local path. This API will move the temporary file, so after a successful call, tempFilePath will no longer be available.

Parameter and description: Object object

Property	Type	Default value	Required	Description
tempFilePath	string	-	True	The path of the temporary file.
filePath	string	-	False	The path where the file will be saved.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.success callback function parameter: Object res

Property	Type	Description
savedFilePath	number	The path of the saved file.

object.fail callback function parameter: Object res

Property	Type	Description
errMsg	string	Error message.

Valid values for res.errMsg

Value	Description
fail tempFilePath file not exist	The specified tempFilePath could not be found.
fail permission denied, open "${filePath}"	No write permission for the specified filePath.

fail no such file or directory "${dirPath}"	The parent directory does not exist.

.saveFileSync

This method is called using number FileSystemManager.saveFileSync(string tempFilePath, string filePath).

Feature description: Synchronously performs the same feature as FileSystemManager.saveFile.

Parameter and description:

string tempFilePath: The path of the temporary file.

string filePath: The path to the file to be saved.

Return value:number savedFilePath. The path where the file will be saved.

.stat

This method is called using FileSystemManager.stat(Object object).

Feature description: Gets the Stats object of a file.

Parameter and description: Object object

Property	Type	Default value	Required	Description
path	string	-	True	The path of the file or directory.
recursive	boolean	false	False	Whether to recursively get the Stats information of each file in the directory.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.success callback function parameter: Object res

Property	Type	Description
stats	Stats/Object	When recursive is false, res.stats is a Stats object. When recursive is true and path is a directory, res.stats is an Object where the key is the relative path from the root

		path and the value is the corresponding Stats object.

object.fail callback function parameter: Object res

Property	Type	Description
errMsg	string	Error message.

Valid values for res.errMsg

Value	Description
fail permission denied, open ${path}	The specified path does not have read permissions.
fail no such file or directory ${path}	The file does not exist.

Example:

When recursive is false:

```
// Asynchronous version
let fs = wx.getFileSystemManager()
fs.stat({
  path: `${wx.env.USER_DATA_PATH}/testDir`,
  success: res => {
    console.log(res.stats.isDirectory())
  }
})
// Synchronous version
fs.statSync(`${wx.env.USER_DATA_PATH}/testDir`, false)
```

When recursive is true:

```
let fs = wx.getFileSystemManager()
// Asynchronous version
fs.stat({
  path: `${wx.env.USER_DATA_PATH}/testDir`,
  recursive: true,
  success: res => {
    Object.keys(res.stats).forEach(path => {
      let stats = res.stats[path]
      console.log(path, stats.isDirectory())
    })
  }
})
// Synchronous version
```

```
fs.statSync(`${wx.env.USER_DATA_PATH}/testDir`, true)
```

.statSync

This method is called using Stats|Object FileSystemManager.statSync(string path, boolean recursive).

Feature description: Synchronously performs the same feature as FileSystemManager.stat.

Parameter and description:

string path: The path to the file/directory.

boolean recursive: Whether to recursively get the stats of each file in the directory.

Return value: Stats|Object stats. When recursive is false, res.stats is a Stats object. When recursive is true and path is a directory, res.stats is an Object where the key is the relative path from the root path and the value is the corresponding Stats object.

Example:

When recursive is false:

```
// Asynchronous version
let fs = wx.getFileSystemManager()
fs.stat({
  path: `${wx.env.USER_DATA_PATH}/testDir`,
  success: res => {
    console.log(res.stats.isDirectory())
  }
})
// Synchronous version
fs.statSync(`${wx.env.USER_DATA_PATH}/testDir`, false)
```

When recursive is true:

```
let fs = wx.getFileSystemManager()
// Asynchronous version
fs.stat({
  path: `${wx.env.USER_DATA_PATH}/testDir`,
  recursive: true,
  success: res => {
    Object.keys(res.stats).forEach(path => {
      let stats = res.stats[path]
      console.log(path, stats.isDirectory())
    })
  }
})
// Synchronous version
```

```
fs.statSync(`${wx.env.USER_DATA_PATH}/testDir`, true)
```

.truncate

This method is called using FileSystemManager.truncate(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Truncates the file content.

Parameter and description: Object object

Property	Type	Default value	Required	Description
filePath	string	-	True	Path to the file to be truncated (local path).
length	number	0	False	The truncation position. Default value: 0. If the length is less than the file length (in bytes), only the first length bytes will be remained in the file, and the rest will be deleted. If the length is greater than the file length, the file will be expanded with empty bytes ('\\0').
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
const fs = wx.getFileSystemManager()
fs.truncate({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  length: 10, // Truncate from the 10th byte
  success(res) {
    console.log(res)
  }
})
```

.truncateSync

This method is called using undefined FileSystemManager.truncateSync(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Synchronously truncates file contents.

Parameter and description: Object object

Property	Type	Default value	Required	Description
filePath	string		True	Path to the file to be truncated (local path).
length	number	0	False	The truncation position. Default value: 0. If the length is less than the file length (in bytes), only the first length bytes will be remained in the file, and the rest will be deleted. If the length is greater than the file length, the file will be expanded with empty bytes ('\\0').

Return value:undefined.

Example:

```
const fs = wx.getFileSystemManager()
fs.truncateSync({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  length: 10, // Truncate from the 10th byte
})
```

.unlink

This method is called using FileSystemManager.unlink(Object object).

Feature description: Deletes a file.

Parameter and description: Object object

Property	Type	Default value	Required	Description
filePath	string	-	True	Path to the file to be deleted.
success	function	-	False	Callback function for successful API calls.

fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.fail callback function parameter: Object res

Property	Type	Description
errMsg	string	Error message.

Valid values for res.errMsg

Value	Description
fail permission denied, open ${path}	The specified path does not have read permissions.
fail no such file or directory ${path}	The file does not exist.
fail operation not permitted, unlink ${filePath}	The provided filePath is a directory.

Example:

```
const fs = wx.getFileSystemManager()
fs.unlink({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  success(res) {
    console.log(res)
  },
  fail(res) {
    console.error(res)
  }
})

// Synchronous API
try {
  const res = fs.unlinkSync(`${wx.env.USER_DATA_PATH}/hello.txt`)
  console.log(res)
} catch(e) {
  console.error(e)
}
```

.unlinkSync

This method is called using FileSystemManager.unlinkSync(string filePath).

Feature description: Synchronously performs the same feature as FileSystemManager.unlink .

Parameter and description:string filePath: Path to the file to be deleted.

Example:

```
const fs = wx.getFileSystemManager()
fs.unlink({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  success(res) {
    console.log(res)
  },
  fail(res) {
    console.error(res)
  }
})

// Synchronous API
try {
  const res = fs.unlinkSync(`${wx.env.USER_DATA_PATH}/hello.txt`)
  console.log(res)
} catch(e) {
  console.error(e)
}
```

.unzip

This method is called using FileSystemManager.unzip(Object object).

Feature description: Decompresses a file.

Parameter and description: Object object

Property	Type	Default value	Required	Description
zipFilePath	string	-	True	Source file path, must be a zip file.
targetPath	string	-	True	The path to the destination directory.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.

complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.fail callback function parameter: Object res

Property	Type	Description
errMsg	string	Error message.

Valid values for res.errMsg

Value	Description
fail permission denied, unzip ${zipFilePath} -> ${destPath}	No write permission for the destination file path.
fail no such file or directory, unzip ${zipFilePath} -> ${destPath}	Source file does not exist, or the parent directory of the destination file path does not exist.

Example:

```
const fs = wx.getFileSystemManager()
fs.unzip({
  zipFilePath: `${wx.env.USER_DATA_PATH}/example.zip`,
  targetPath: '${wx.env.USER_DATA_PATH}/example',
  success(res) {
    console.log(res)
  },
  fail(res) {
    console.error(res)
  }
})
```

.write

This method is called using FileSystemManager.write(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Writes to a file.

Parameter and description: Object object

Property	Type	Default	Required	Description

		value		
fd	string	-	True	File descriptor obtained through FileSystemManager.open or FileSystemManager.openSync.
data	string/ArrayBuffer	-	True	The text or binary data to be written.
offset	number	0	False	Effective only when data is of type ArrayBuffer. Specifies the index in the ArrayBuffer to start writing from. Default is 0.
length	number	-	False	Effective only when data is of type ArrayBuffer. Specifies the number of bytes to write. Default is the remaining bytes from offset.
encoding.	string	utf8	False	Character encoding for writing to the file.
position	number	-	False	Specifies the offset from the beginning of the file where data should be written. If not provided or not a number, data will be written at the current pointer position.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Valid values for encoding

Value	Description
ascii	-
base64	-
binary	-
hex	-
ucs2/ucs-2/utf16le/utf-16le	Reads in little-endian order.
utf-8/utf8	-
latin1	-

object.success callback function parameter: Object res

Property	Type	Description
bytesWritten	number	The actual number of bytes written to the file (note that the number of bytes written may not match the number of characters in the string).

Example:

```
const fs = wx.getFileSystemManager()
// Open a file
fs.open({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  flag: 'a+',
  success(res) {
    // Write to the file
    fs.write({
      fd: res.fd,
      data: 'some text',
      success(res) {
        console.log(res.bytesWritten)
      }
    })
  }
})
```

.writeSync

This method is called using FileSystemManager.writeFile(Object object).

Note:

This API is supported in mini programs but not in mini games.

Feature description: Synchronously writes to a file.

Parameter and description: Object object

Property	Type	Default value	Required	Description
fd	string	-	True	File descriptor obtained through FileSystemManager.open or FileSystemManager.openSync.
data	string/ArrayBuffer	-	True	The text or binary data to be written.

offset	number	0	False	Effective only when data is of type ArrayBuffer. Specifies the index in the ArrayBuffer to start writing from. Default is 0.
length	number	-	False	Effective only when data is of type ArrayBuffer. Specifies the number of bytes to write. Default is the remaining bytes from offset.
encoding	string	utf8	False	Character encoding for writing to the file.
position	number	-	False	Specifies the offset from the beginning of the file where data should be written. If not provided or not a number, data will be written at the current pointer position.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Valid values for encoding

Value	Description
ascii	-
base64	-
binary	-
hex	-
ucs2/ucs-2/utf16le/utf-16le	Reads in little-endian order.
utf-8/utf8	-
latin1	-

object.success callback function parameter: Object res

Property	Type	Description
bytesWritten	number	The actual number of bytes written to the file (note that the number of bytes written may not match the number of characters in the string).

Example:

```
const fs = wx.getFileSystemManager()
const fd = fs.openSync({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  flag: 'a+'
})
const res = fs.writeSync({
  fd: fd,
  data: 'some text'
})
console.log(res.bytesWritten)
```

.writeFile

This method is called using FileSystemManager.writeFile(Object object).

Feature description: Writes a file.

Parameter and description: Object object

Property	Type	Default value	Required	Description
filePath	string	-	True	The path to the file to be written.
data	string/ArrayBuffer	-	True	The text or binary data to be written.
encoding.	string	utf8	False	Character encoding for writing to the file.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Valid values for object.encoding

Value	Description
ascii	-
base64	-
binary	-

hex	-
ucs2/ucs-2/utf16le/utf-16le	Reads in little-endian order.
utf-8/utf8	-
latin1	-

object.fail callback function parameter: Object res

Property	Type	Description
errMsg	string	Error message.

Valid values for res.errMsg

Value	Description
fail no such file or directory, open ${filePath}	The specified filePath directory does not exist.
fail permission denied, open ${dirPath}	No write permission for the specified filePath.

Example:

```
const fs = wx.getFileSystemManager()
fs.writeFile({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  data: 'some text or arrayBuffer',
  encoding: 'utf8',
  success(res) {
    console.log(res)
  },
  fail(res) {
    console.error(res)
  }
})

// Synchronous API
try {
  const res = fs.writeFileSync(
    `${wx.env.USER_DATA_PATH}/hello.txt`,
    'some text or arrayBuffer',
    'utf8'
  )
  console.log(res)
} catch(e) {
```

```
  console.error(e)
}
```

.writeFileSync

This method is called using FileSystemManager.writeFileSync(string filePath, string|ArrayBuffer data, string encoding).

Feature description: Synchronously performs the same feature as FileSystemManager.writeFile.

Parameter and description:

string filePath: The path to the file to be written.

string|ArrayBuffer data: The text or binary data to be written.

string encoding: The character encoding for writing to the file.

Valid values for encoding

Value	Description
ascii	-
base64（When using base64 encoding, only pass the base64 content itself. Do not include the Data URI prefix, otherwise a fail base64 encode error will occur. For example, pass aGVsbG8= instead of ）	-
binary	-
hex	-
ucs2/ucs-2/utf16le/utf-16le	Reads in little-endian order.
utf-8/utf8	-
latin1	-

Example:

```
const fs = wx.getFileSystemManager()
fs.writeFile({
  filePath: `${wx.env.USER_DATA_PATH}/hello.txt`,
  data: 'some text or arrayBuffer',
  encoding: 'utf8',
  success(res) {
    console.log(res)
```

```
  },
  fail(res) {
    console.error(res)
  }
})

// Synchronous API
try {
  const res = fs.writeFileSync(
    `${wx.env.USER_DATA_PATH}/hello.txt`,
    'some text or arrayBuffer',
    'utf8'
  )
  console.log(res)
} catch(e) {
  console.error(e)
}
```

Error codes

Note:

Unless otherwise specified, the error codes are subject to the following table.

Error code	Error message.	Description
1300001	operation not permitted	The operation is not allowed (e.g., it is expected to provide a file for filePath but actually a directory is provided).
1300002	no such file or directory ${path}	File/directory does not exist, or the parent directory is missing.
1300005	Input/output error	Input/output stream is unavailable.
1300009	bad file descriptor	Invalid file descriptor.
1300013	permission denied	Permission error. The file is read-only or write-only.
1300014	Path permission denied	No permission for the specified path.
1300020	not a directory	The specified path is not a directory.
1300021	Is a directory	The specified path is a directory.
1300022	Invalid argument	Invalid argument, check if length or offset is out of range.
1300036	File name too long	The filename is too long.
1300066	directory not empty	The directory is not empty.

1300201	system error	The system API is failed to be called.
1300202	the maximum size of the file storage limit is exceeded	Storage space is insufficient, or file size exceeds the limit (100 MB).
1300203	base64 encode error	Character encoding conversion failed (e.g., incorrect base64 format).
1300300	sdcard not mounted	Failed to mount the Android SDCard.
1300301	unable to open as fileType	Unable to open the file as the specified fileType.
1301000	permission denied, cannot access file path	No access to the target path (e.g., usr directory)
1301002	data to write is empty	The data to be written is empty.
1301003	illegal operation on a directory	Illegal operation on a directory (e.g., specified filePath is an existing directory).
1301004	illegal operation on a package directory	Illegal operation on a code package directory.
1301005	file already exists ${dirPath}	A file or directory with the same name already exists.
1301006	value of length is out of range	The length value is out of range.
1301007	value of offset is out of range	The offset value is out of range.
1301009	value of position is out of range	The position value is out of range.
1301100	store directory is empty	The store directory is empty.
1301102	unzip open file fail	Failed to open the compressed file.
1301103	unzip entry fail	Failed to unzip a single file.
1301104	unzip fail	Unzipping failed.
1301111	brotli decompress fail	Brotli decompression failed (e.g., the specified compressionAlgorithm does not match the actual compression format of the file).
1301112	tempFilePath file not exist	The specified tempFilePath could not be found.
1302001	fail permission denied	No read/write permission for the specified fd path.
1302002	exceed max concurrent fd limit	Maximum number of file descriptors reached.

1302003	invalid flag	Invalid flag.
1302004	permission denied when open using flag	Unable to open the file using the specified flag.
1302005	array buffer does not exist	arrayBuffer is not provided.
1302100	array buffer is readonly	The arrayBuffer is read-only.

Stats

An object that describes the status of a file.

Property

number mode: The type and access permissions of the file, corresponding to POSIX stat.st_mode.
number size: The file size in bytes, corresponding to POSIX stat.st_size.
number lastAccessedTime: The last time the file was accessed or executed, in UNIX timestamp format, corresponding to POSIX.
number lastModifiedTime: The last time the file was modified, in UNIX timestamp format, corresponding to POSIX stat.st_mtime.

Methods

.isDirectory

This method is called using boolean Stats.isDirectory().

Feature description: Checks whether the current file is a directory.
Return value:boolean, indicating whether the current file is a directory.

.isFile

This method is called using boolean Stats.isFile().

Feature description: Checks whether the current file is a normal file.
Return value:boolean, indicating whether the current file is a normal file.

ReadResult

Feature description: The result of reading the files. It is returned using FileSystemManager.readSync.
Property:
number bytesRead: The actual number of bytes read.
ArrayBuffer arrayBuffer: The buffer object into which the data was read, which is the same as the arrayBuffer passed as an input parameter to the API.

WriteResult

Feature description: The result of writing to a file. It is returned using FileSystemManager.writeSync.
Property:number bytesWritten: The actual number of bytes written to the file (note that the number of bytes written may not match the number of characters in the string).

©2013-2025 Tencent Cloud International Pte. Ltd.

Device
Contact

Last updated：2024-11-21 18:48:11

chooseContact

This API is used via wx.chooseContact(Object object).

Feature Description: Initiates the mobile phone's contact list, allowing the selection of a contact.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success callback function: Object.

Attribute	Type	Description
phoneNumber	string	Mobile number
displayName	string	Contact name
phoneNumberList	string	All mobile numbers of the selected contact (some Android systems only allow selecting contacts, not specific phone numbers)

addPhoneContact

This API is used via wx.addPhoneContact(Object object).

Feature Description: Adds mobile phone contacts. Users can choose to write this form into the mobile phone system's contact list by either **adding a new contact** or **adding to an existing contact**.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
firstName	string	-	Yes	Name
photoFilePath	string	-	No	Local file path of avatar
nickName	string	-	No	Nickname
lastName	string	-	No	Surname
middleName	string	-	No	Middle Name
remark	string	-	No	Remarks
mobilePhoneNumber	string	-	No	Mobile number
weChatNumber	string	-	No	WeChat ID
addressCountry	string	-	No	Country of address
addressState	string	-	No	Province of address
addressCity	string	-	No	City of address
addressStreet	string	-	No	Sub-district of address
addressPostalCode	string	-	No	Postcode
organization	string	-	No	Company
title	string	-	No	Position
workFaxNumber	string	-	No	Office fax number
workPhoneNumber	string	-	No	Office telephone number
hostNumber	string	-	No	Company phone number
email	string	-	No	Email Address
url	string	-	No	Website
workAddressCountry	string	-	No	Country of office address

workAddressState	string	-	No	Province of office address
workAddressCity	string	-	No	City of office address
workAddressStreet	string	-	No	Sub-district of office address
workAddressPostalCode	string	-	No	Postcode of office address
homeFaxNumber	string	-	No	Residential Fax
homePhoneNumber	string	-	No	Residential phone number
homeAddressCountry	string	-	No	Country of Residential Address
homeAddressState	string	-	No	Province of Residential Address
homeAddressCity	string	-	No	City of Residential Address
homeAddressStreet	string	-	No	Sub-district of residential address
homeAddressPostalCode	string	-	No	Postcode of residential address
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

©2013-2025 Tencent Cloud International Pte. Ltd.

Vibration

Last updated：2024-11-21 18:48:11

vibranteShort

This API is used via wx.vibrateShort(Object object).

Feature Description: Triggers a brief vibration (15 ms) on the mobile device. It is only effective on iPhone 7/7 Plus and newer models, as well as on Android devices.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

vibranteLong

This API is used via wx.vibrateLong(Object object).

Feature Description: Triggers a prolonged vibration on the mobile phone (400 ms).

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the

				interface invocation (both successful and unsuccessful invocations)

Battery

Last updated：2025-02-25 14:21:58

getBatteryInfoSync

The utilization method for this API is Object wx.getBatteryInfoSync().

Function Description: The synchronous version of wx.getBatteryInfo.

Return Value: Object res.

Attributes	Types	Note
level	number	Device battery level, ranging from 1 - 100.
isCharging	boolean	Is the device currently charging?

getBatteryInfo

The method for utilizing this API is wx.getBatteryInfo(Object object).

Function Description: This function retrieves the device's battery level. The synchronous API wx.getBatteryInfoSync is not available on iOS.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

©2013-2025 Tencent Cloud International Pte. Ltd.

object.success Callback Function Parameters: Object res.

Attributes	Types	Note
level	number	Device battery level, ranging from 1 - 100.
isCharging	boolean	Is the device currently charging?

Screen

Last updated：2025-02-24 17:10:36

setVisualEffectOnCapture

Notes:

This API is supported in mini programs but not in mini games.

This API is called using wx.setVisualEffectOnCapture(Object object).

Feature description: Sets the screen display when taking screenshots or recording the screen.(for Android only).

Parameter and description: Object object.

Property	Type	Default value	Required	Description
visualEffect	string	none	False	The visual effect during screenshot/recording. Supports none or hidden. Passing hidden will hide the screen during capture.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

setScreenBrightness

This API is called using wx.setScreenBrightness(Object object).

Feature description: Sets the screen brightness.

Parameter and description: Object object.

Property	Type	Default value	Required	Description

value	number	-	True	Screen brightness value, ranging from 0 (darkest) to 1 (brightest). On Android, supports a special value of -1, which means the screen brightness follows the system setting.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

setKeepScreenOn

This API is called using wx.setKeepScreenOn(Object object).

Feature description: Sets whether to keep the screen on. This setting only takes effect within the current mini program and will be invalid once you leave the mini program.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
keepScreenOn	boolean	-	True	Whether to keep the screen on.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
wx.setKeepScreenOn({
  keepScreenOn: true
})
```

onUserCaptureScreen

This API is called using wx.onUserCaptureScreen(function listener).

Notes:

This API is supported in mini programs but not in mini games.

Feature description: Listens for user-initiated screenshot event. It is triggered when the user takes a screenshot using the system screenshot button. Only one listener can be registered.

Parameter and description:function listener. The listener for user-initiated screenshot event.

Example:

```
wx.onUserCaptureScreen(function (res) {
  console.log('User took a screenshot')
})
```

onScreenRecordingStateChanged

Notes:

This API is supported in mini programs but not in mini games.

This API is called using wx.onScreenRecordingStateChanged(function listener).

Feature description: Listens for user-initiated screen recording event.

Parameter and description:function listener. The listener for user-initiated screen recording event.

Property	Type	Description
state	string	Screen recording status. Valid values: start: Screen recording started stop: Screen recording stopped

Example:

```
// Listen for user-initiated screen recording event
const handler = function (res) {
  console.log(res.state)
}
wx.onScreenRecordingStateChanged(handler)

// Cancel listening for user screen recording event
```

```
wx.offScreenRecordingStateChanged(handler)
```

offUserCaptureScreen

This API is called using wx.offUserCaptureScreen(function callback).

Notes:

This API is supported in mini programs but not in mini games.

Feature description: Cancels the listener for user-initiated screenshot event.

Parameter and description:function callback. The callback function for user-initiated screenshot event.

offScreenRecordingStateChanged

Notes:

This API is supported in mini programs but not in mini games.

This API is called using wx.offScreenRecordingStateChanged(function listener).

Feature description: Removes the listener for user-initiated screen recording event.

Parameter and description:function listener. The listener passed to onScreenRecordingStateChanged. If not provided, all listeners will be removed.

Example:

```
// Listen for user-initiated screen recording event
const handler = function (res) {
  console.log(res.state)
}
wx.onScreenRecordingStateChanged(handler)

// Cancel listening for user screen recording event
wx.offScreenRecordingStateChanged(handler)
```

```
const listener = function (res) { console.log(res) }

wx.onScreenRecordingStateChanged(listener)
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
wx.offScreenRecordingStateChanged(listener) // Must pass the same function object u
```

getScreenRecordingState

Notes:

This API is supported in mini programs but not in mini games.

This API is called using wx.getScreenRecordingState(Object object).

Feature description: Queries whether the user is currently recording the screen.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.success callback function parameter: Object res.

Property	Type	Description
state	string	Screen recording status. Valid values: on: Enabled off: Disabled

Example:

```
wx.getScreenRecordingState({
  success: function (res) {
    console.log(res.state)
  },
})
```

getScreenBrightness

This API is called using wx.getScreenBrightness(Object object).

Notes:

If the automatic brightness adjustment feature is enabled in the Android system settings, the screen brightness will automatically adjust according to the light. This API can only obtain the value before the automatic brightness adjustment, not the real-time brightness value.

Feature description: Gets the screen brightness.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

object.success callback function parameter: Object object.

Property	Type	Description
value	number	Screen brightness value, ranging from 0 (darkest) to 1 (brightest).

Memory

Last updated：2024-11-21 18:48:11

onMemoryWarning

This API is used via wx.onMemoryWarning(function listener).

Feature Description: Monitors low memory warning events. When iOS/Android issues a memory warning to the mini program process, these events are triggered. The triggering of such events does not imply that the mini program process has been cleared. In most cases, it is merely a warning. Developers can recycle some unnecessary resources to avoid further exacerbating memory tension after receiving the notification.
Parameter and Description: function listener, the monitoring function for low memory warning events.

Attribute	Type	Description
level	number	Memory warning level, only available on Android, corresponds to system macro definition.

Valid Values for Level

Valid Values	Description
5	TRIM_MEMORY_RUNNING_MODERATE
10	TRIM_MEMORY_RUNNING_LOW
15	TRIM_MEMORY_RUNNING_CRITICAL

Sample Code

```
wx.onMemoryWarning(function () {
  console.log('onMemoryWarningReceive')
})
```

offMemoryWarning

This API is used via wx.offMemoryWarning(function listener).

Feature Description: Removes the listener function for low memory warning events.

Parameters and Description: function listener, the listener function passed in by onMemoryWarning. If this parameter is not passed in, all listener functions will be removed.

Sample Code

```
const listener = function (res) { console.log(res) }

wx.onMemoryWarning(listener)
wx.offMemoryWarning(listener) // The same function object as the listener must be p
```

Device Motion

Last updated：2025-02-25 11:55:10

stopDeviceMotionListening

Notes:

This API is supported in mini programs but not in mini games.

This API is called using wx.stopDeviceMotionListening(Object object).

Feature description: Stops listening for device motion change event.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

startDeviceMotionListening

Notes:

This API is supported in mini programs but not in mini games.

This API is called using wx.startDeviceMotionListening(Object object).

Feature description: Starts listening for device motion change event.

Parameter and description: Object object.

Property	Type	Default value	Required	Description

©2013-2025 Tencent Cloud International Pte. Ltd.

interval	string	normal	False	The execution frequency of the callback function that listens for device orientation changes. Valid values: game: Callback frequency suitable for game updates, about 20 ms/time. ui: Callback frequency suitable for UI updates, about 60 ms/time. normal: Normal callback frequency, about 200 ms/time.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

onDeviceMotionChange

This API is called using wx.onDeviceMotionChange(function listener).

Feature description: Listens for device motion change event. The frequency is based on the interval parameter of the wx.startDeviceMotionListening(). You can use wx.stopDeviceMotionListening() to stop listening.

Parameter and description:function listener. The listener for device orientation change event.

Property	Type	Description
alpha	number	The angle around the Z-axis when the phone's X/Y coordinates align with the Earth's X/Y coordinates. The range is [0, 2*PI). Counterclockwise rotation is positive.
beta	number	The angle around the X-axis when the phone's Y/Z coordinates align with the Earth's Y/Z coordinates. The range is [-1*PI, PI). Rotation towards the Earth's surface is positive. It may also be positive towards the user.
gamma	number	The angle around the Y-axis when the phone's X/Z coordinates align with the Earth's X/Z coordinates. The range is [-1*PI/2, PI/2). Rotation towards the Earth's surface on the right side is positive.

offDeviceMotionChange

This API is called using wx.offDeviceMotionChange(function listener).

©2013-2025 Tencent Cloud International Pte. Ltd.

Feature description: Removes the listener for device orientation change event.

Parameter and description:function listener. The listener passed to onDeviceMotionChange. If not provided, all listeners will be removed.

Example:

```
const listener = function (res) { console.log(res) }

wx.onDeviceMotionChange(listener)
wx.offDeviceMotionChange(listener) // Must pass the same function object used in on
```

©2013-2025 Tencent Cloud International Pte. Ltd.

Accessibility

Last updated：2024-11-21 18:48:11

checkIsOpenAccessibility

This API is used via wx.checkIsOpenAccessibility(Object object).

Feature Description: Detects whether the visual accessibility feature has been activated.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function		No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object.

Attribute	Type	Note ·
open	Boolean	Returns "true" when the auxiliary function narrative is enabled on iOS, or when talkback is enabled on Android.

©2013-2025 Tencent Cloud International Pte. Ltd.

Network

Last updated：2024-11-21 18:48:11

onNetworkStatusChange

This API is used via wx.onNetworkStatusChange(function callback).

Feature Description: function callback, a callback function for network state change events.

Parameter and Description: Function callback.

Callback function parameters for network status change events: Object res.

Attribute	Type	Description
isConnected	boolean	Current Network Connection Status
networkType	string	Network type

Legitimate Values of Legitimate

Value	Description
wifi	Wi-Fi Network
2g	2G Network
3g	3G Network
4g	4G Network
unknown	Uncommon Network Types on Android
none	No network

Sample Code

```
wx.onNetworkStatusChange(function (res) {
  console.log(res.isConnected)
  console.log(res.networkType)
})
```

getNetworkType

This API is used via wx.getNetworkType(Object object).

Feature Description: Gets the network type.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Object.success callback function: Object res.

Attribute	Type	Description
networkType	string	Network type

Valid values of res.networkType

Value	Description
wifi	Wi-Fi Network
2g	2G Network
3g	3G Network
4g	4G Network
unknown	Uncommon Network Types on Android
none	No network

Sample Code

```
wx.getNetworkType({
  success(res) {
```

```
    const networkType = res.networkType
  }
})
```

offNetworkStatusChange

This API is used via wx.offNetworkStatusChange(function listener).

Feature Description: Removes the listener function for network status change events.

Parameter and Description: function listener, the listener function passed in by onNetworkStatusChange. If this parameter is not passed in, all listener functions will be removed.

Sample Code

```
const listener = function (res) { console.log(res) }

wx.onNetworkStatusChange(listener)
wx.offNetworkStatusChange(listener) // The same function object as the listener mus
```

Keyboard

Last updated：2024-11-21 18:48:11

onKeyboardHeightChange

This API is used via wx.onKeyboardHeightChange(function listener)

Feature Description: Listen for keyboard height change events.

Parameter and Description: function callback, listener for keyboard height change events.

Listener function parameter: Object res.

Attribute	Type	Description
height	number	Keyboard Height

Sample Code

```
wx.onKeyboardHeightChange(res => {
  console.log(res.height)
})
```

hideKeyboard

This API is used via wx.hideKeyboard(Object object).

Feature Description: Call this interface manually to retract the keyboard after it has been pulled up by the input, textarea, etc. focus.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call

©2013-2025 Tencent Cloud International Pte. Ltd.

complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Sample Code

```
wx.hideKeyboard({
  complete: res => {
    console.log('hideKeyboard res', res)
  }
})
```

offKeyboardHeightChange

This API is used via wx.offKeyboardHeightChange(function listener).

Feature Description: Removes the listener function for keyboard height change events.

Parameter and Description: function listener, the listener function passed in by onKeyboardHeightChange. If this parameter is not passed in, all listener functions will be removed.

Sample Code

```
const listener = function (res) { console.log(res) }

wx.onKeyboardHeightChange(listener)
wx.offKeyboardHeightChange(listener) // The same function object as the listener mu
```

getSelectTextRange

This API is used via wx.getSelectedTextRange(Object object).

Feature Description: After input, textarea, etc. are focused, the cursor position in the input box is obtained. **This interface is only effective when called during focus.**

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description

success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Object.success callback function: Object res.

Attribute	Type	Description
start	number	Initial cursor position in the input field
end	number	Final cursor position in the input field

Sample Code

```
wx.getSelectedTextRange({
  complete: res => {
    console.log('getSelectedTextRange res', res.start, res.end)
  }
})
```

Phone Call

Last updated：2025-02-25 16:53:06

makePhonecall

This API is used via wx.makePhoneCall(Object object).

Feature Description: Initiates a phone call.

Parameter: Object.

Attribute	Type	Default value	Required	Description
phoneNumber	string	-	Yes	The phone number that needs to be dialed
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Sample Code

```
wx.makePhoneCall({
  phoneNumber: '1340000' // This is merely an example and not an actual phone numbe
})
```

Accelerometer

Last updated：2025-02-24 17:10:36

startAccelerometer

This API is used via wx.startAccelerometer(Object object).

Note:

Depending on the performance of the device, as well as the current CPU and memory usage, there may be some discrepancies between the set `interval` and the actual execution frequency of the wx.onAccelerometerChange callback function.

Feature Description: Starts monitoring acceleration data.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
interval	string	normal	No	Execution frequency of the callback function for monitoring acceleration data
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Valid values of object.interval

Value	Description
game	Suitable for updating the callback frequency of the game, approximately every 20 ms.
ui	Suitable for updating the callback frequency of the UI, approximately every 60 ms.
normal	Regular callback frequency, approximately every 200 ms.

Sample Code

```
wx.startAccelerometer({
  interval: 'game'
})
```

stopAccelerometer

This API is used via wx.stopAccelerometer(Object object).

Feature Description: Stops monitoring acceleration data.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Sample Code

```
wx.stopAccelerometer()
```

onAccelerometerChange

This API is used via wx.onAccelerometerChange(function listener).

Feature Description: Monitor acceleration data events. The frequency depends on the interval parameter of wx.startAccelerometer(). The monitoring can be stopped using wx.stopAccelerometer().

Parameter and Description: Function callback.

The callback function for acceleration data events:

Attribute	Type	Description

x	number	X-axis
y	number	Y-axis
z	number	Z-axis

Sample Code

```
wx.onAccelerometerChange(function (res) {
  console.log(res.x)
  console.log(res.y)
  console.log(res.z)
})
```

offAccelerometerChange

This API is used via wx.offAccelerometerChange(function listener).

Feature Description:

Parameters and Description: function listener, the listening function passed into onAccelerometerChange. If this parameter is not provided, all listening functions will be removed.

Sample Code

```
const listener = function (res) { console.log(res) }

wx.onAccelerometerChange(listener)
wx.offAccelerometerChange(listener) // The same function object as the listener mus
```

©2013-2025 Tencent Cloud International Pte. Ltd.

Compass

Last updated：2025-02-24 17:10:36

startCompass

This API is called using wx.startCompass(Object object).

Feature description: Starts listening for compass data.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example:

```
wx.startCompass()
```

stopCompass

This API is called using wx.stopCompass(Object object).

Feature description: Stops listening for compass data.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls.

©2013-2025 Tencent Cloud International Pte. Ltd.

fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Example

```
wx.stopCompass()
```

onCompassChange

This API is called using wx.onCompassChange(function listener).

Feature description: Listens for compass data change event. Frequency: 5 times per second. Automatically starts listening after the API call, and can be stopped using wx.stopCompass.
Parameter and description:function listener. The callback function for compass data change event. Parameter: Object res.

Property	Type	Description
direction	number	The direction in degrees.
accuracy	number/string	The accuracy.

Example

```
wx.onCompassChange(function (res) {
    console.log(res.direction)
})
```

The accuracy difference on iOS and Android: Due to platform differences, the accuracy has different values on iOS and Android.
iOS: The accuracy is is a number representing the deviation from magnetic north. 0 means the device points to magnetic north, 90 means east, 180 means south, and so on.
Android: accuracy is a string representing an enumerated value.

Value	Description
high	High accuracy.

medium	Medium accuracy.
low	Low accuracy.
no-contact	Unreliable. The sensor connection is lost.
unreliable	Unreliable for unknown reasons.
unknow ${value}	Unknown accuracy enumeration value, meaning the value returned by the Android system is not a standard accuracy enumeration value.

offCompassChange

This API is called using wx.offCompassChange(function listener).

Feature description: Removes the listener for compass data change event.

Parameter and description:function listener. The listener passed to onCompassChange. If not provided, all listeners will be removed.

Example:

```
const listener = function (res) { console.log(res) }

wx.onCompassChange(listener)
wx.offCompassChange(listener) // Must pass the same function object used in onCompa
```

Gyroscope

Last updated：2025-02-24 17:10:36

startGyroscope

This API is called using wx.startGyroscope(Object object).

Feature description: Starts listening for gyroscope data.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
interval	string	normal	False	The execution frequency of the callback function that listens for gyroscope data.
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

Valid values for object.interval:

Value	Description
game	Callback frequency suitable for game updates, about 20 ms/time.
ui	Callback frequency suitable for UI updates, about 60 ms/time.
normal	Normal callback frequency, about 200 ms/time.

stopGyroscope

This API is called using wx.stopGyroscope(Object object).

©2013-2025 Tencent Cloud International Pte. Ltd.

Feature description: Stops listening for gyroscope data.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls.
fail	function	-	False	Callback function for failed API calls.
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure).

onGyroscopeChange

This API is called using wx.onGyroscopeChange(function listener).

Feature description: Listens for gyroscope data change event. The frequency is based on the interval parameter of the wx.startGyroscope(). You can use wx.stopGyroscope() to stop listening.

Parameter and description:function listener. The callback function for gyroscope data change event. Parameter: Object res.

Property	Type	Description
x	number	The angular velocity around the x-axis.
y	number	The angular velocity around the y-axis.
z	number	The angular velocity around the z-axis.

offGyroscopeChange

This API is called using wx.offGyroscopeChange(function listener).

Feature description: Removes the listener for gyroscope data change event.

©2013-2025 Tencent Cloud International Pte. Ltd.

Parameter and description:function listener. The listener passed to onGyroscopeChange. If not provided, all listeners will be removed.

Example

```
const listener = function (res) { console.log(res) }

wx.onGyroscopeChange(listener)
wx.offGyroscopeChange(listener) // Must pass the same function object used in onGyr
```

SMS

Last updated：2024-11-21 18:48:11

sendSms

This API is used via wx.sendSms(Object object).

Feature Description: Initiates the mobile interface for sending SMS.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
phoneNumber	string	-	No	Mobile number pre-filled in the SMS sending panel
content	string	-	No	Pre-filled content in the SMS sending panel
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

©2013-2025 Tencent Cloud International Pte. Ltd.

Code Scanning

Last updated：2025-02-24 17:10:36

scanCode

This API is called using wx.scanCode(Object object).

Notes:

This API is supported in mini programs but not in mini games.

Feature description: Opens the client's code scanning interface to scan the code.

Parameter and description: Object object.

Property	Type	Valid values and description	Default value	Required	Description
onlyFromCamera	boolean	-	false	False	Whether to only allow scanning from the camera, not from the photo album.
scanType	Array.<string>	barCode: Barcode qrCode: QR code datamatrix: Data Matrix code pdf417: PDF417 barcode	['barCode', 'qrCode']	False	Types of codes to scan.
success	function	-	-	False	Callback function for successful API calls.
fail	function	-	-	False	Callback function for failed API calls.
complete	function	-	-	False	Callback function executed after API call ends (regardless of success or failure).

©2013-2025 Tencent Cloud International Pte. Ltd.

object.success callback function parameter: Object res.

Property	Type	Valid values and description	Description
result	string	-	Content of the scanned code.
scanType	string	QR_CODE: QR code AZTEC: Barcode CODABAR: Barcode CODE_39: Barcode CODE_93: Barcode CODE_128: Barcode DATA_MATRIX: QR code EAN_8: Barcode EAN_13: Barcode ITF: Barcode MAXICODE: Barcode PDF_417: QR code RSS_14: Barcode RSS_EXPANDED: Barcode UPC_A: Barcode UPC_E: Barcode UPC_EAN_EXTENSION: Barcode WX_CODE: QR code CODE_25: Barcode	Type of the scanned code.
charSet	string	-	Character set of the scanned code.
path	string	-	If the scanned code is a QR code for the current mini program, this field will be returned with the path carried by the QR code
rawData	string	-	Base64-encoded raw data.

Example:

```
// Allow scanning from both camera and photo album
wx.scanCode({
  success (res) {
    console.log(res)
  }
})

// Only allow scanning from the camera
wx.scanCode({
```

```
  onlyFromCamera: true,
  success (res) {
    console.log(res)
  }
})
```

Encryption

Last updated：2024-11-21 18:48:11

getRandomValues

This API is used via wx.getRandomValues(Object object).

Feature Description: Gets cryptographically secure random numbers.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
length	number	-	Yes	Integer, the byte size of the generated random number, with a maximum of 1048576.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
randomValues	ArrayBuffer	Random number content, with a length equivalent to the passed-in byte size.

Sample Code

```
wx.getRandomValues({
  length: 6 // Generate a random number of 6-byte length,
  success: res => {
    console.log(wx.arrayBufferToBase64(res.randomValues)) // Print after converting
  }
})
```

NFC—Ndef

Last updated：2024-11-21 18:48:11

.connect

The method is utilized as Ndef.connect().

Function Description: Establishes a connection with the NFC tag.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.close

The method is employed as Ndef.close().

Function Description: Disconnects the link.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not	Callback function for unsuccessful interface

			required	invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

©2013-2025 Tencent Cloud International Pte. Ltd.

.setTimeout

The method is utilized as Ndef.setTimeout(Object object).

Function Description: Setting the timeout duration.

Parameters and Description: Object object

Attributes	Types	Default value	Required	Note
timeout	number	-	Supported	Setting the timeout duration (ms)
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.

©2013-2025 Tencent Cloud International Pte. Ltd.

13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.isConnected

The method is utilized as Ndef.isConnected().

Note:

This interface has been deprecated; developers can maintain the connection status independently.

Function Description: Verifies if a connection has been established.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error	Error Message	Note

Codes		
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.offNdefMessage

The method is utilized as Ndef.onNdefMessage(function callback).

Function Description: Cancels the monitoring of Ndef messages.

Parameters and Description: function callback.

.onNdefMessage

The method is employed as Ndef.offNdefMessage(function callback).

Function Description: Monitors Ndef messages.

Parameters and Description: function callback.

.writeNdefMessage

The method is employed as Ndef.writeNdefMessage(Object object).

Function Description: Overwrites the content of Ndef tags.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
uris	Array	-	Not required	URI array.
texts	Array	-	Not required	Text array.
records	Array	-	Not required	An array of binary objects, requiring specification of id, type, and payload (all of which are of the ArrayBuffer type).
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error	Error Message	Note

Codes		
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

©2013-2025 Tencent Cloud International Pte. Ltd.

NFC—NfcA

Last updated：2024-11-21 18:48:11

.connect

The method is utilized as NfcA.connect().

Function Description: Establishes a connection with the NFC tag.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed at the conclusion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.

©2013-2025 Tencent Cloud International Pte. Ltd.

13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.close

The method is utilized as NfcA.close().

Function Description: Disconnect.

Parameters and Description: Object object

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.setTimeout

> The method is employed as NfcA.setTimeout(Object object).

Function Description: Setting the timeout duration.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
timeout	number	-	Supported	Setting the timeout duration (ms)
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.

©2013-2025 Tencent Cloud International Pte. Ltd.

13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.isConnected

The method is utilized as NfcA.isConnected().

Note:

This interface has been deprecated; developers can maintain the connection status independently.

Function Description: Verifies if a connection has been established.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied

13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.getMaxTransceiveLength

The method is utilized as NfcA.getMaxTransceiveLength().

Function Description: Retrieves the maximum transmission length.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and

				unsuccessful invocations)

object.success Callback Function Parameters: Object res.

Attributes	Types	Note
length	number	Maximum Transmission Length

Error Codes

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.

13016	connect fail	Connection failed

.transceive

The method is employed as NfcA.transceive(Object object).

Function Description: Transmits data.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
data	ArrayBuffer	-	Supported	Binary data required for transmission
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success Callback Function Parameters: Object res.

Attributes	Types	Note
data	ArrayBuffer	-

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied

13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.getAtqa

The method is utilized as NfcA.getAtqa().

Function Description: Retrieves ATQA information.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation

complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success Callback Function Parameters: Object res.

Attributes	Types	Note
atqa	ArrayBuffer	Returns ATQA/SENS_RES data.

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.

13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.getSak

The method is employed as NfcA.getSak().

Function Description: Acquires SAK information.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success Callback Function Parameters: Object res.

Attributes	Types	Note
sak	number	Returns SAK/SEL_RES data.

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied

13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

©2013-2025 Tencent Cloud International Pte. Ltd.

NFC—NfcB

Last updated：2024-11-21 18:48:12

.connect

The method is utilized via NfcB.connect().

Function Description: Establishes a connection with the NFC tag.

Parameters and Description: Object - object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.

©2013-2025 Tencent Cloud International Pte. Ltd.

13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.close

The method is utilized as NfcB.close().

Function Description: Disconnects the link.

Parameters and Description: Object object

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.setTimeout

The method is employed as NfcB.setTimeout(Object object).

Function Description: Setting the timeout duration.

Parameters and Description: Object object

Attributes	Types	Default value	Required	Note
timeout	number	-	Supported	Setting the timeout duration (ms)
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology

©2013-2025 Tencent Cloud International Pte. Ltd.

		from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.isConnected

The method is utilized as NfcB.isConnected().

Function Description: Verifies if a connection has been established.

Parameters and Description: Object - object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter

13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.getMaxTransceiveLength

The method is utilized as NfcB.getMaxTransceiveLength().

Function Description: Retrieves the maximum transmission length.

Parameters and Description: Object - object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful

				invocations)

object.success Callback Function Parameters: Object res.

Attributes	Types	Note
length	number	Maximum Transmission Length

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.transceive

The method is employed as NfcB.transceive(Object object).

Function Description: Transmits data.

Parameters and Description: Object - object.

Attributes	Types	Default value	Required	Note
data	ArrayBuffer	-	Supported	Binary data required for transmission
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success Callback Function Parameters: Object res.

Attributes	Types	Note
data	ArrayBuffer	-

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter

13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

NFC—NfcF

Last updated：2024-11-21 18:48:11

.connect

The method is utilized via NfcF.connect().

Function Description: Establishes a connection with the NFC tag.

Parameters and Description: Object - object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.

©2013-2025 Tencent Cloud International Pte. Ltd.

13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.close

The method is utilized as NfcF.close().

Function Description: Disconnects the link.

Parameters and Description: Object - object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.setTimeout

The method is employed as NfcF.setTimeout(Object object).

Function Description: Setting the timeout duration.

Parameters and Description: Object - object.

Attributes	Types	Default value	Required	Note
timeout	number	-	Supported	Setting the timeout duration (ms).
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.

©2013-2025 Tencent Cloud International Pte. Ltd.

13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.isConnected

The method is utilized as NfcF.isConnected().

Function Description: Verifies if a connection has been established.

Parameters and Description: Object - object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-

13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.getMaxTransceiveLength

The method is employed as NfcF.getMaxTransceiveLength().

Function Description: Retrieves the maximum transmission length.

Parameters and Description: Object - object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation

complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success Callback Function Parameters: Object res.

Attributes	Types	Note
length	number	Maximum Transmission Length

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.

13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.transceive

The method is utilized as NfcF.transceive(Object object).

Function Description: Transmits data.

Parameters and Description: Object - object.

Attributes	Types	Default value	Required	Note
data	ArrayBuffer	-	Supported	Binary data required for transmission
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success Callback Function Parameters: Object res.

Attributes	Types	Note
data	ArrayBuffer	-

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-

13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

©2013-2025 Tencent Cloud International Pte. Ltd.

NFC—NfcV

Last updated：2024-11-21 18:48:11

.connect

The method is utilized via NfcV.connect().

Function Description: Establishes a connection with the NFC tag.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.

©2013-2025 Tencent Cloud International Pte. Ltd.

13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.close

The method is utilized as NfcV.close().

Function Description: Disconnects the link.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.setTimeout

The method is employed as NfcV.setTimeout(Object object).

Function Description: Setting the timeout duration.

Parameters and Description: Object object

Attributes	Types	Default value	Required	Note
timeout	number	-	Supported	Setting the timeout duration (ms)
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.

13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.isConnected

The method is utilized as NfcV.isConnected()

Note:

This interface has been deprecated; developers can maintain the connection status independently.

Function Description: Verifies if a connection has been established.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-

13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.getMaxTransceiveLength

The method is employed as NfcV.getMaxTransceiveLength()

Function Description: Retrieves the maximum transmission length.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation

complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success Callback Function Parameters: Object res.

Attributes	Types	Note
length	number	Maximum Transmission Length

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.

©2013-2025 Tencent Cloud International Pte. Ltd.

13016	connect fail	Connection failed

.transceive

The method is utilized as NfcV.transceive(Object object)

Function Description: Transmits data.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
data	ArrayBuffer	-	Supported	Binary data required for transmission
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success Callback Function Parameters: Object res.

Attributes	Types	Note
data	ArrayBuffer	-

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-

13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

©2013-2025 Tencent Cloud International Pte. Ltd.

WiFi

Last updated：2025-03-25 18:15:55

onWifiConnected

The method of utilizing this API is wx.onWifiConnected(function listener).

Note:

Compatibility:

Android: Supported.

iOS: Supported.

Function Description: Monitors the event of connecting to a Wi-Fi network.

Parameters and Description: function listener, the listener function for the event of connecting to Wi-Fi, with the following arguments Object res:

Attributes	Types	Note
wifi	WifiInfo	Wi-Fi Information

offWifiConnected

The method of utilizing this API is wx.onWifiConnected(function listener).

Function Description: Monitors the event of connecting to a Wi-Fi network.

Parameters and Description: function listener, onWifiConnected The listener function passed in. Not passing this parameter removes all listener functions.

Attributes	Types	Note
wifi	WifiInfo	Wi-Fi Information

Sample code

```
const listener = function (res) { console.log(res) }
```

©2013-2025 Tencent Cloud International Pte. Ltd.

```
wx.onWifiConnected(listener)
wx.offWifiConnected(listener) // Pass in the same function object as the listener.
```

stopWifi

The usage method for this API is wx.stopWifi(Object object)

Function Description: Deactivates the Wi-Fi module.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
0	ok	Normal
12000	not init	The startWifi interface was not invoked beforehand.
12001	system not support	The current system does not support the relevant capabilities.
12002	password error Wi-Fi	The password is incorrect.
12003	connection timeout	Connection timeout, supported only on Android.
12004	duplicate request	Repeated connection to Wi-Fi.
12005	wifi not turned on	Exclusive to Android, the Wi-Fi switch has not been turned on.

12006	gps not turned on	Exclusive to Android, the GPS location switch has not been turned on.
12007	user denied	The user has denied authorization to connect to Wi-Fi.
12008	invalid SSID	Invalid SSID.
12009	system config err	The system operator configuration has denied the connection to Wi-Fi.
12010	system internal error	System miscellaneous error, it is necessary to print the specific error reason in errmsg.
12011	weapp in background	The application is unable to configure Wi-Fi while running in the background.
12013	wifi config may be expired	The system's saved Wi-Fi configuration has expired. It is recommended to forget the Wi-Fi and retry. This is only supported on Android.
12014	invalid WEP / WPA password	Specific to iOS, the WEP/WPA password is invalid.

Example Code

```
wx.stopWifi({
  success (res) {
    console.log(res.errMsg)
  }
})
```

startWifi

The usage method for this API is wx.startWifi(Object object).

Function Description: Initializes the Wi-Fi module.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not	Callback function for successful interface invocation

			required	
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
0	ok	Normal
12000	not init	The startWifi interface was not invoked beforehand.
12001	system not support	The current system does not support the relevant capabilities.
12002	password error Wi-Fi	The password is incorrect.
12003	connection timeout	Connection timeout, supported only on Android.
12004	duplicate request	Repeated connection to Wi-Fi.
12005	wifi not turned on	Exclusive to Android, the Wi-Fi switch has not been turned on.
12006	gps not turned on	Exclusive to Android, the GPS location switch has not been turned on.
12007	user denied	The user has denied authorization to connect to Wi-Fi.
12008	invalid SSID	Invalid SSID.
12009	system config err	The system operator configuration has denied the connection to Wi-Fi.
12010	system internal error	System miscellaneous error, it is necessary to print the specific error reason in errmsg.
12011	weapp in background	The application is unable to configure Wi-Fi while running in the background.
12013	wifi config may be expired	The system's saved Wi-Fi configuration has expired. It is recommended to forget the Wi-Fi and retry. This is only supported on Android.

12014	invalid WEP / WPA password	Specific to iOS, the WEP/WPA password is invalid.

Example Code

```
wx.startWifi({
  success (res) {
    console.log(res.errMsg)
  }
})
```

onGetWifiList

The usage method for this API is wx.onGetWifiList(function listener).

Function Description: Monitors the event of receiving Wi-Fi list data.

Parameters and Description: Object res parameter, function listener, the monitoring function for the event of receiving Wi-Fi list data.

Attributes	Types	Note
wifiList	Array.<WifiInfo>	Wi-Fi List Data

offGetWifiList

The usage method for this API is wx.offGetWifiList(function listener).

Function Description: Removes the monitoring function for the event of receiving Wi-Fi list data.

Parameters and Description: Function listener, the monitoring function passed into onGetWifiList. If this parameter is not provided, all monitoring functions will be removed.

Sample code

```
const listener = function (res) { console.log(res) }

wx.onWifiConnected(listener)
wx.offWifiConnected(listener) // Pass in the same function object as the listener.
```

©2013-2025 Tencent Cloud International Pte. Ltd.

getConnectedWifi

The usage method for this API is wx.getConnectedWifi(Object object).

Function Description: Retrieves information about the currently connected Wi-Fi.

Parameters and Description: Object object

Attributes	Types	Default value	Required	Note
partialInfo	boolean	false	Not required	Whether partial Wi-Fi information needs to be returned.
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success Callback Function Parameters: Object res

Attributes	Types	Note
wifi	WifiInfo	Wi-Fi Information

Error Code

Error Codes	Error Message	Note
0	ok	Normal
12000	not init	The startWifi interface was not invoked beforehand.
12001	system not support	The current system does not support the relevant capabilities.
12002	password error Wi-Fi	The password is incorrect.

12003	connection timeout	Connection timeout, supported only on Android.
12004	duplicate request	Repeated connection to Wi-Fi.
12005	wifi not turned on	Exclusive to Android, the Wi-Fi switch has not been turned on.
12006	gps not turned on	Exclusive to Android, the GPS location switch has not been turned on.
12007	user denied	The user has denied authorization to connect to Wi-Fi.
12008	invalid SSID	Invalid SSID.
12009	system config err	The system operator configuration has denied the connection to Wi-Fi.
12010	system internal error	System miscellaneous error, it is necessary to print the specific error reason in errmsg.
12011	weapp in background	The application is unable to configure Wi-Fi while running in the background.
12013	wifi config may be expired	The system's saved Wi-Fi configuration has expired. It is recommended to forget the Wi-Fi and retry. This is only supported on Android.
12014	invalid WEP / WPA password	Specific to iOS, the WEP/WPA password is invalid.

WifiInfo

Function Description: Wi-Fi Information.

Note:

For Android wx.connectWifi / wx.getConnectedWifi, if partialInfo:true is set, or if the wx.onWifiConnectedWithPartialInfo event is called, a WifiInfo object containing only the SSID property will be returned.

For iOS wx.getConnectedWifi, if partialInfo:true is set, a WifiInfo object containing only the SSID and BSSID properties will be returned. However, the user must enable the host client's location permissions for the results to be correctly returned.

In certain circumstances, even though the Wi-Fi may have successfully connected, an error may occur due to the inability to obtain a complete WifiInfo object. The specific error messages are `errCode:12010, errMsg: can't gain current wifi` or `no wifi is connected`. If developers do not require a complete WifiInfo object, they can resolve the error issue by adopting the aforementioned strategy.

©2013-2025 Tencent Cloud International Pte. Ltd.

Attributes

string SSID: The SSID of the Wi-Fi.

string BSSID: The BSSID of the Wi-Fi.

boolean secure: The security status of the Wi-Fi.

number signalStrength: The signal strength of the Wi-Fi. For Android, the value ranges from 0 to 100, while for iOS, the value ranges from 0 to 1. The larger the value, the stronger the signal strength.

number frequency: The frequency band of the Wi-Fi, measured in MHz.

©2013-2025 Tencent Cloud International Pte. Ltd.

Calendar

Last updated：2024-11-21 18:48:11

addPhoneCalendar

This API is used via wx.addPhoneCalendar(Object object)

Feature Description: Adds events into the system calendar.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
title	string	-	Yes	Calendar Event Title
startTime	number	-	Yes	Unix timestamp of the start time
allDay	boolean	-	No	Whether it is an all-day event, which defaults to "false".
description	string	-	No	Event Description
location	string	-	No	Event Location
endTime	string	-	No	Unix timestamp of the end time
alarm	boolean	-	No	Whether to send a reminder, which defaults to "true".
alarmOffset	number	-	No	Reminder lead time, in seconds. The default value is 0, indicating a reminder at the start of the event.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

addPhoneRepeatCalendar

This API is used via wx.addPhoneRepeatCalendar(Object object)

Feature Description: Adds recurring events into the system calendar.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
title	string	-	Yes	Calendar Event Title
startTime	number	-	Yes	Unix timestamp of the start time (number of seconds elapsed since January 1, 1970)
allDay	boolean	-	No	Whether it is an all-day event, which defaults to "false".
description	string	-	No	Event Description
location	string	-	No	Event Location
endTime	string	-	No	Unix timestamp of the end time
alarm	boolean	-	No	Whether to send a reminder, which defaults to "true".
alarmOffset	number	-	No	Reminder lead time, in seconds. The default value is 0, indicating a reminder at the start of the event.
repeatInterval	string	-	No	Repetition cycle, defaulting to 'month' for monthly repetition
repeatEndTime	number	-	No	Unix timestamp for the end of the repetition cycle. If left blank, it implies indefinite repetition.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

repeatInterval

Valid Values	Description
day	Recurring day by day

week	Recurring week by week
month	Recurring month by month. Under this mode, the date should not exceed the 28th day.
year	Recurring year by year

©2013-2025 Tencent Cloud International Pte. Ltd.

Clipboard

Last updated：2024-11-21 18:48:11

setClipboardData

This API is used via wx.setClipboardData(Object object).

Feature Description: Sets the content of the system clipboard. For individual users, it should not be invoked more than once per second.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
data	string	-	Yes	Clipboard content
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Sample Code

```
wx.setClipboardData({
  data: 'data',
  success(res) {
    wx.getClipboardData({
      success(res) {
        console.log(res.data) // data
      }
    })
  }
})
```

getClipboardData

This API is used via wx.getClipboardData(Object object).

Feature Description: Gets the content of the system clipboard. For individual users, the invocation does not exceed `1` time per second.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success callback function: Object.

Attribute	Type	Description
data	string	Clipboard content

Sample Code

```
wx.getClipboardData({
  success(res) {
    console.log(res.data)
  }
})
```

NFC—getNFCAdapter

Last updated：2024-11-21 18:48:11

The application of this API is through the NFCAdapter wx.getNFCAdapter().

Note:

Compatibility:

Android Version: Supported.

iOS Version: Not Supported.

Function Description: Acquire NFC Instance.

Parameters and Description: Returns NFCAdapter NFC Instance.

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User has not granted permission.
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding

©2013-2025 Tencent Cloud International Pte. Ltd.

		technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

©2013-2025 Tencent Cloud International Pte. Ltd.

NFC—NFCAdapter

Last updated：2025-03-25 18:15:56

NFCAdapter

Parameters and Description: Object tech, enumeration of tag types.

Attributes	Types	Note
ndef	String	Corresponds to the Ndef instance, the instance supports reading and writing of NDEF data on NFC tags in NDEF format.
nfcA	String	Corresponds to the NfcA instance, the instance supports reading and writing in accordance with the NFC-A (ISO 14443-3A) standard.
nfcB	String	Corresponds to the NfcB instance, the instance supports reading and writing in accordance with the NFC-B (ISO 14443-3B) standard.
isoDep	String	Corresponds to the IsoDep instance, the instance supports reading and writing in accordance with the ISO-DEP (ISO 14443-4) standard.
nfcF	String	Corresponds to the NfcF instance, the instance supports reading and writing in accordance with the NFC-F (JIS 6319-4) standard.
nfcV	String	Corresponds to the NfcV instance, the instance supports reading and writing in accordance with the NFC-V (ISO 15693) standard.
mifareClassic	String	Corresponds to the MifareClassic instance, the instance supports reading and writing of MIFARE Classic tags.
mifareUltralight	String	Corresponds to the MifareUltralight instance, the instance supports reading and writing of MIFARE Ultralight tags.

.startDiscovery

The method is utilized as NFCAdapter.startDiscovery().

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	Function	-	Not	Callback function for successful interface

			required	invocation
fail	Function	-	Not required	Callback function for successful interface invocation
complete	Function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.

13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.stopDiscovery

The method is utilized as NFCAdapter.stopDiscovery().

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.

©2013-2025 Tencent Cloud International Pte. Ltd.

13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.getNdef

The method is employed as Ndef NFCAdapter.getNdef().

Function Description: Acquire the Ndef instance, which supports reading and writing of NDEF data on NFC tags in NDEF format.
Return Value: Ndef.

.getNfcA

The method is implemented as NfcA NFCAdapter.getNfcA().

Function Description: Obtain the NfcA instance, which supports reading and writing in accordance with the NFC-A (ISO 14443-3A) standard.
Return Value: NfcA.

.getNfcB

The method is utilized as NfcB NFCAdapter.getNfcB().

Function Description: Acquire the NfcB instance, which facilitates reading and writing in compliance with the NFC-B (ISO 14443-3B) standard.
Return Value: NfcB.

.getNfcF

The method is employed as NfcF NFCAdapter.getNfcF().

Function Description: Procure the NfcF instance, which supports reading and writing in accordance with the NFC-F (JIS 6319-4) standard.
Return Value: NfcF.

.getNfcV

The method is utilized as NfcV NFCAdapter.getNfcV().

Function Description: Acquire the NfcV instance, which supports reading and writing in compliance with the NFC-V (ISO 15693) standard.
Return Value: NfcV.

.getIsoDep

The method is employed as IsoDep NFCAdapter.getIsoDep().

Function Description: Obtain the IsoDep instance, which supports reading and writing in accordance with the ISO-DEP (ISO 14443-4) standard.
Return Value: IsoDep.

.getMifareClassic

The method is utilized as MifareClassic NFCAdapter.getMifareClassic().

Function Description: Acquire the MifareClassic instance, which supports reading and writing of MIFARE Classic tags.

©2013-2025 Tencent Cloud International Pte. Ltd.

Return Value: MifareClassic.

.getMifareUltralight

The method is employed as MifareUltralight NFCAdapter.getMifareUltralight().

Function Description: Obtain the MifareUltralight instance, which supports the reading and writing of MIFARE Ultralight tags.
Parameter Description: MifareUltralight.

.onDiscovered

The method is utilized as NFCAdapter.onDiscovered(function listener).

Function Description: Monitor NFC Tag.
Parameter Description:function listener, listener function, with the following arguments Object res:

Attributes	Types	Note
id	ArrayBuffer	-
techs	Array	Tech array, used to match which specific standards (such as NfcA instances) can be used to handle NFC cards.
messages	Array	NdefMessage array, the message format is {id: ArrayBuffer, type: ArrayBuffer, payload: ArrayBuffer}.

.offDiscovered

The method is utilized as NFCAdapter.offDiscovered(function listener).

Function Description: Remove the listener function of the NFC Tag.
Parameters and Description: function listener, the listening function passed in by onDiscovered. If this parameter is not passed, all listening functions will be removed.
Example Code

```
const listener = function (res) { console.log(res) }
```

```
NFCAdapter.onDiscovered(listener)
NFCAdapter.offDiscovered(listener) // The same function object as the listener must
```

©2013-2025 Tencent Cloud International Pte. Ltd.

NFC—IsoDep

Last updated：2024-11-21 18:48:12

.connect

The method is utilized as IsoDep.connect().

Function Description: Establishes a connection with the NFC tag.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback for interface invocation failure.
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC.	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.

©2013-2025 Tencent Cloud International Pte. Ltd.

13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.close

The method is employed as IsoDep.close().

Function Description: Disconnects the link.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC.	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.setTimeout

The method is implemented as IsoDep.setTimeout(Object object).

Function Description: Setting the timeout duration.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
timeout	number	-	Supported	Setting the timeout duration (ms).
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function		Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC.	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.

©2013-2025 Tencent Cloud International Pte. Ltd.

13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.isConnected

The method is utilized as IsoDep.isConnected().

Function Description: Verifies if a connection has been established.

Parameters and Description: Object object

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC.	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.

13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.getMaxTransceiveLength

The method is executed as IsoDep.getMaxTransceiveLength().

Function Description: Retrieves the maximum transmission length.

Parameters and Description: Object object

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success Callback Function Parameters: Object res

Attributes	Types	Note

length	number	Maximum Transmission Length

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC.	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.transceive

The method is executed as IlsoDep.transceive(Object object).

Function Description: Transmits data.

Parameters and Description: Object object

Attributes	Types	Default value	Required	Note
data	ArrayBuffer	-	Supported	Binary data required for transmission
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success Callback Function Parameters: Object res

Attributes	Types	Note
data	ArrayBuffer	-

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC.	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.

13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.getHistoricalBytes

The method is invoked as IsoDep.getHistoricalBytes().

Function Description: Acquiring reset information.

Parameters and Description: Object object

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success Callback Function Parameters: Object res

Attributes	Types	Note
histBytes	ArrayBuffer	Returns historical binary data.

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC.	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

©2013-2025 Tencent Cloud International Pte. Ltd.

NFC—MifareClassic

Last updated：2024-11-21 18:48:12

.connect

The method is utilized as MifareClassic.connect().

Function Description: Establishes a connection with the NFC tag.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.

13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.close

The method is employed as MifareClassic.close().

Function Description: Disconnects the link.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error	Error Message	Note

Codes		
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.setTimeout

The method is implemented as MifareClassic.setTimeout(Object object).

Function Description: Setting the timeout duration.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
timeout	number	-	Supported	Setting the timeout duration (ms).
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.

©2013-2025 Tencent Cloud International Pte. Ltd.

13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.isConnected

The method is invoked as MifareClassic.isConnected().

Note:

This interface has been deprecated; developers can maintain the connection status independently.

Function Description: Verifies if a connection has been established.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied

13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.getMaxTransceiveLength

The method is executed as MifareClassic.getMaxTransceiveLength().

Function Description: Acquiring the Maximum Transmission Length.

Parameters and Description: Object object

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success Callback Function Parameters: Object res

Attributes	Types	Note
length	number	Maximum Transmission Length

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.transceive

The method is utilized as MifareClassic.transceive(Object object).

Function Description: Transmitting data.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
data	ArrayBuffer	-	Supported	Binary data required for transmission
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success Callback Function Parameters: Object res

Attributes	Types	Note
data	ArrayBuffer	-

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.

13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

©2013-2025 Tencent Cloud International Pte. Ltd.

NFC—MifareUltralight

Last updated：2024-11-21 18:48:11

.connect

The method is utilized as MifareUltralight.connect().

Function Description: Establishes a connection with the NFC tag.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for failed interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.

©2013-2025 Tencent Cloud International Pte. Ltd.

13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.close

The method is utilized as MifareUltralight.close().

Function Description: Disconnects the link.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-

13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.setTimeout

The method is employed as MifareUltralight.setTimeout(Object object).

Function Description: Setting the timeout duration.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note

timeout	number	-	Supported	Setting the timeout duration (ms)
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

©2013-2025 Tencent Cloud International Pte. Ltd.

.isConnected

The method is utilized as MifareUltralight.isConnected().

Note:

This interface has been deprecated; developers can maintain the connection status independently.

Function Description: Verifies if a connection has been established.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.

13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.getMaxTransceiveLength

The method is utilized as MifareUltralight.getMaxTransceiveLength().

Function Description: Retrieves the maximum transmission length.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success Callback Function Parameters: Object res

Attributes	Types	Note
length	number	Maximum Transmission Length

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.
13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

.transceive

The method is employed as MifareUltralight.transceive(Object object).

Function Description: Transmits data.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
data	ArrayBuffer	-	Supported	Binary data required for transmission
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success Callback Function Parameters: Object res.

Attributes	Types	Note
data	ArrayBuffer	-

Error Code

Error Codes	Error Message	Note
13000	The device does not support NFC	-
13001	The system's NFC switch is not activated.	-
13010	An unknown error occurred.	-
13019	user is not authorized	User Authorization Denied
13011	invalid parameter	Invalid parameter
13012	parse NdefMessage failed	Failed to parse the parameter into NdefMessage.
13021	NFC discovery already started	NFC scanning has already commenced.
13018	NFC discovery has not started	Attempted to halt NFC scanning without initiating it.
13022	Tech already connected	The tag has been successfully connected.

13023	Tech has not connected	Attempted to disconnect without a tag being connected.
13013	NFC tag has not been discovered	No NFC tag was detected.
13014	invalid tech	Invalid tag technology.
13015	unavailable tech	Failed to acquire the corresponding technology from the tag.
13024	function not support	The current tag technology does not support this feature.
13017	system internal error	The related read-write operation has failed.
13016	connect fail	Connection failed

©2013-2025 Tencent Cloud International Pte. Ltd.

Bluetooth - Low Energy Central Device

Last updated：2024-11-21 18:48:12

writeBLECharacteristicValue

This API is used via wx.writeBLECharacteristicValue(Object object).

Note:

Simultaneous multiple calls may potentially result in write failures.

While the mini program does not impose restrictions on the size of the data packet written, the system and Bluetooth device will limit the data size of a single Bluetooth 4.0 transmission. Since exceeding the maximum byte count may result in write errors, it is recommended not to exceed 20 bytes per write operation.

If data is written in a single instance that is excessively long, iOS may not provide any callbacks, including error callbacks.

On the Android platform, invoking this interface immediately after successfully calling wx.notifyBLECharacteristicValueChange may result in a system error coded 10008 on certain models.

Feature Description: Writes binary data into the characteristic value of a low-energy Bluetooth device. **Successful invocation requires the device's characteristic to support write operations**.

Parameter and Description: Object.

Attribute	Type	Required	Description
deviceId	string	Yes	Bluetooth Device ID
serviceId	string	Yes	UUID corresponding to the Bluetooth characteristic service
characteristicId	string	Yes	Bluetooth characteristic UUID
value	ArrayBuffer	Yes	The binary data of the characteristic value from the Bluetooth device.
writeType	string	No	Write mode setting for Bluetooth feature value, there are two modes, iOS priority write and Android priority writeNoResponse. legal value is: write: Mandatory write response, reporting error when unsupported. 'writeNoResponse': Mandatory write without response, reporting error when unsupported.

success	function	No	Callback Function of Successful Interface Call
fail	function	No	Callback Function of Successful Interface Call
complete	function	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error code	Error Message	Description
0	ok	Normal
-1	already connect	Connected
10000	not init	Bluetooth Adapter Not Initialized
10001	not available	The current Bluetooth adapter is unavailable.
10002	no device	Specified Device Not Found
10003	connection fail	Connection failed.
10004	no service	Specified Service Not Found
10005	no characteristic	Specified characteristic not found
10006	no connection	The current connection has been disconnected.
10007	property not support	The current characteristic does not support this operation.
10008	system error	Exceptions Reported by All Other Systems
10009	system not support	Exclusive to Android system, and BLE is not supported on versions earlier than 4.3.
10012	operate time out	Connection timeout
10013	invalid_data	The connection deviceId is either null or incorrectly formatted.

Sample Code

```
// Transmit a hexadecimal data of 0x00 to the Bluetooth device.
let buffer = new ArrayBuffer(1)
let dataView = new DataView(buffer)
dataView.setUint8(0, 0)

wx.writeBLECharacteristicValue({
```

```
  // The deviceId here must be obtained through the getBluetoothDevices or onBlueto
  deviceId,
  // The ServiceId here must be obtained from the wx.getBLEDeviceServices interface
  serviceId,
  // The CharacteristicId Here must be obtained from the getBLEDeviceCharacteristic
  characteristicId,
  // The value here is of ArrayBuffer type.
  value: buffer,
  success (res) {
    console.log('writeBLECharacteristicValue success', res.errMsg)
  }
})
```

readBLECharacteristicValue

This API is used via wx.readBLECharacteristicValue(Object object).

Note:

Simultaneous multiple calls may potentially result in read failures.

The Information read from the interface must be obtained in the callback registered by the wx.onBLECharacteristicValueChange method.

Feature Description: Reads the binary data of the low-energy Bluetooth device characteristics. **Successful invocation is only possible if the device's characteristic supports read**.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
deviceId	string	-	Yes	Bluetooth Device ID
serviceId	string	-	Yes	UUID corresponding to the Bluetooth characteristic service
characteristicId	string	-	Yes	Bluetooth characteristic UUID
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error code	Error Message	Description
0	ok	Normal
-1	already connect	Connected
10000	not init	Bluetooth Adapter Not Initialized
10001	not available	The current Bluetooth adapter is unavailable.
10002	no device	Specified Device Not Found
10003	connection fail	Connection failed.
10004	no service	Specified Service Not Found
10005	no characteristic	Specified characteristic not found
10006	no connection	The current connection has been disconnected.
10007	property not support	The current characteristic does not support this operation.
10008	system error	Exceptions Reported by All Other Systems
10009	system not support	Exclusive to Android system, and BLE is not supported on versions earlier than 4.3.
10012	operate time out	Connection timeout
10013	invalid_data	The connection deviceId is either null or incorrectly formatted.

Sample Code

```
// Retrieval is only possible in this callback.
wx.onBLECharacteristicValueChange(function(characteristic) {
  console.log('characteristic value comed:', characteristic)
})

wx.readBLECharacteristicValue({
  // The deviceId here must have already established a link with the corresponding
  deviceId,
  // The ServiceId here must be obtained from the wx.getBLEDeviceServices interface
  serviceId,
  // The CharacteristicId Here Must be Retrieved from the getBLEDeviceCharacteristi
```

```
  characteristicId,
  success (res) {
    console.log('readBLECharacteristicValue:', res.errCode)
  }
})
```

setBLEMTU

This API is used via wx.setBLEMTU(Object object).

Feature Description: Negotiates the setting of the maximum transmission unit (MTU) for low-energy Bluetooth devices. It needs to be called after wx.createBLEConnection has been successfully invoked. This is only effective on Android systems version 5.1 and later versions, and is not supported on iOS due to system limitations.
Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
deviceId	string	-	Yes	Bluetooth Device ID
mtu	number	-	Yes	Maximum Transmission Unit. The setting range is within the interval (22,512), measured in bytes.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
mtu	number	The final negotiated MTU value, which is consistent with the parameter passed in. It is supported from Android client 8.0.9 onwards.

getBLEMTU

This API is used via wx.getBLEMTU(Object object).

Note:

Within the mini program, the MTU is ATT_MTU, which includes the length of the Op-Code and Attribute Handle. The actual data transmission length is ATT_MTU - 3.

In the iOS system, the MTU is a fixed value; in the Android system, the MTU changes after successful system negotiation. It is recommended to use wx.onBLEMTUChange for monitoring.

Feature Description: Gets the maximum transmission unit of low-energy Bluetooth devices. It needs to be called after wx.createBLEConnection has been successfully invoked.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
deviceId	string	-	Yes	Bluetooth Device ID
writeType	string	write	No	Write mode (specific to iOS). Valid values are: write: Write with response. writeNoResponse: Write without response.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Successful Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
mtu	number	Maximum Transmission Unit

Error Code

Error code	Error Message	Description
0	ok	Normal
-1	already connect	Connected
10000	not init	Bluetooth Adapter Not Initialized

10001	not available	The current Bluetooth adapter is unavailable.
10002	no device	Specified Device Not Found
10003	connection fail	Connection failed.
10004	no service	Specified Service Not Found
10005	no characteristic	Specified characteristic not found
10006	no connection	The current connection has been disconnected.
10007	property not support	The current characteristic does not support this operation.
10008	system error	Exceptions Reported by All Other Systems
10009	system not support	Exclusive to Android system, and BLE is not supported on versions earlier than 4.3.
10012	operate time out	Connection timeout
10013	invalid_data	The connection deviceId is either null or incorrectly formatted.

Sample Code

```
wx.getBLEMTU({
  deviceId: '',
  writeType: 'write',
  success (res) {
    console.log(res)
  }
})
```

onBLEMTUChange

This API is used via wx.onBLEMTUChange(function listener).

Feature Description: Monitors events of MTU changes in low-energy Bluetooth devices (only triggered on Android).
Parameter and Description: Object res parameter, function listener, the listener function for events of MTU changes in low-energy Bluetooth devices.

Attribute	Type	Description

deviceId	string	Bluetooth Device ID
mtu	number	Maximum Transmission Unit

Sample Code

```
wx.onBLEMTUChange(function (res) {
  console.log('bluetooth mtu is', res.mtu)
})
```

offBLEMTUChange

This method is used via wx.offBLEMTUChange(function listener).

Feature Description: Removes the listener function for events of MTU changes in low-energy Bluetooth devices.
Parameter and Description: function listener, the listener function passed in by onBLEMTUChange. If this parameter is not passed in, all listener functions will be removed.
Sample Code

```
const listener = function (res) { console.log(res) }

wx.onBLEMTUChange(listener)
wx.offBLEMTUChange(listener) // The same function object as the listener must be pa
```

onBLEConnectionStateChange

This API is used via wx.onBLEConnectionStateChange(function listener).

Feature Description: Monitors events of changes to low-energy Bluetooth connection status. This includes active connections or disconnections by developers, device loss, exceptional disconnections, and so on.
Parameters and Description: function listener, the Bluetooth low power connection state change event listener function, the parameter Object res is as follows:

Attribute	Type	Description
deviceId	string	Bluetooth Device ID

connected	boolean	Whether it is in connected status.

Sample Code

```
wx.onBLEConnectionStateChange(function(res) {
  // This method callback can be used to handle exceptional situations such as unex
  console.log(device ${res.deviceId} state has changed, connected: ${res.connected}
})
```

offBLEConnectionStateChange

This API is used via wx.offBLEConnectionStateChange(function listener).

Feature Description: Removes the listener function for events of changes to low-energy Bluetooth connection status.
Parameter and Description: function listener, the listener function passed in by onBLEConnectionStateChange. If this parameter is not passed in, all listener functions will be removed.
Sample Code

```
const listener = function (res) { console.log(res) }

wx.onBLEConnectionStateChange(listener)
wx.offBLEConnectionStateChange(listener) // The same function object as the listene
```

onBLECharacteristicValueChange

This API is used via wx.onBLECharacteristicValueChange(function listener).

Feature Description: Monitors events of characteristic value changes for low-energy Bluetooth devices. You must first call the wx.notifyBLECharacteristicValueChange interface to receive notifications pushed by the device.
Parameter and Description: function listener, the listener function for the feature value change event of the Bluetooth low power device, with the following parameters Object res:

Attribute	Type	Description
deviceId	string	Bluetooth Device ID

serviceId	string	UUID corresponding to the Bluetooth characteristic service
characteristicId	string	Bluetooth characteristic UUID
value	ArrayBuffer	The latest characteristic value

Sample Code

```
// Example of converting ArrayBuffer to hexadecimal string
function ab2hex(buffer) {
  let hexArr = Array.prototype.map.call(
    new Uint8Array(buffer),
    function(bit) {
      return ('00' + bit.toString(16)).slice(-2)
    }
  )
  return hexArr.join('');
}
wx.onBLECharacteristicValueChange(function(res) {
  console.log(characteristic ${res.characteristicId} has changed, now is ${res.valu
  console.log(ab2hex(res.value))
})
```

offBLECharacteristicValueChange

This API is used via wx.offBLECharacteristicValueChange().

Feature Description: Removes all listener functions for events of characteristic value changes for low-energy Bluetooth devices.

Sample Code

```
wx.offBLECharacteristicValueChange()
```

notifyBLECharacteristicValueChange

This API is used via wx.notifyBLECharacteristicValueChange(Object object).

Note:

After the subscription operation is successful, the device needs to actively update the characteristic value to trigger the wx.onBLECharacteristicValueChange callback.

On the Android platform, invoking the wx.writeBLECharacteristicValue interface immediately after successfully calling this interface may result in a system error coded 10008 on certain models.

The device's characteristic must support "notify" or "indicate" for the call to be successful. Additionally, wx.notifyBLECharacteristicValueChange must be enabled first to monitor the device's characteristicValueChange event.

Feature Description: Enables the "notify" feature when the characteristic value of low-energy Bluetooth device changes, subscribing to the characteristics.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
deviceId	string	-	Yes	Bluetooth Device ID
serviceId	string	-	Yes	UUID corresponding to the Bluetooth characteristic service
characteristicId	string	-	Yes	Bluetooth characteristic UUID
state	boolean	-	Yes	Whether to enable "notify"
type	string	indication	No	Set the characteristic subscription type, whose valid values include "notification" and "indication".
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error code	Error Message	Description
0	ok	Normal
-1	already connect	Connected
10000	not init	Bluetooth Adapter Not Initialized

10001	not available	The current Bluetooth adapter is unavailable.
10002	no device	Specified Device Not Found
10003	connection fail	Connection failed.
10004	no service	Specified Service Not Found
10005	no characteristic	Specified characteristic not found
10006	no connection	The current connection has been disconnected.
10007	property not support	The current characteristic does not support this operation.
10008	system error	Exceptions Reported by All Other Systems
10009	system not support	Exclusive to Android system, and BLE is not supported on versions earlier than 4.3.
10012	operate time out	Connection timeout
10013	invalid_data	The connection deviceId is either null or incorrectly formatted.

Sample Code

```
// Transmit a hexadecimal data of 0x00 to the Bluetooth device.
let buffer = new ArrayBuffer(1)
let dataView = new DataView(buffer)
dataView.setUint8(0, 0)

wx.writeBLECharacteristicValue({
  // The deviceId here must be obtained through the getBluetoothDevices or onBlueto
  deviceId,
  // The ServiceId here must be obtained from the wx.getBLEDeviceServices interface
  serviceId,
  // The CharacteristicId Here Must be Retrieved from the getBLEDeviceCharacteristi
  characteristicId,
  // The Value Here is of ArrayBuffer Type
  value: buffer,
  success (res) {
    console.log('writeBLECharacteristicValue success', res.errMsg)
  }
})
```

getBLEDeviceServices

This API is used via wx.getBLEDeviceServices(Object object).

Feature Description: Gets all services from the low-energy Bluetooth device.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
deviceId	string	-	Yes	Bluetooth device id. A connection must have already been established through wx.createBLEConnection.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
services	Array.<Object>	Device service list

Structure attribute of service

Attribute	Type	Description
uuid	string	Bluetooth device service UUID
isPrimary	boolean	Whether the service is a primary service.

Sample Code

```
wx.getBLEDeviceServices({
    // The deviceId here must have already established a link with the correspondin
  deviceId,
  success (res) {
    console.log('device services:', res.services)
  }
})
```

getBLEDeviceRSSI

This API is used via wx.getBLEDeviceRSSI(Object object).

Feature Description: Gets the signal strength (Received Signal Strength Indication, RSSI) of the low-energy Bluetooth device.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
deviceId	string	-	Yes	Bluetooth Device ID
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
RSSI	Number	Signal strength, measured in dBm

getBLEDeviceCharacteristics

This API is used via wx.getBLEDeviceCharacteristics(Object object).

Feature Description: Gets all characteristics within a service from a low-energy Bluetooth device.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
deviceId	string	-	Yes	Bluetooth device id. A connection must have already been established through wx.createBLEConnection.

serviceId	string	-	Yes	Bluetooth service UUID. It is necessary to first invoke wx.getBLEDeviceServices to get it.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
characteristics	Array.<Object>	Device characteristic value list

Structure attributes of characteristics

Structure attributes	Type	Description
uuid	string	Bluetooth device characteristic UUID
properties	Object	Operation types supported by this characteristic

Structure attributes of properties

Attribute	Type	Description
read	boolean	Whether this characteristic supports the read operation.
write	boolean	Whether this characteristic supports the write operation.
notify	boolean	Whether this characteristic supports the notify operation.
indicate	boolean	Whether this characteristic supports the indicate operation.
writeNoResponse	boolean	Whether this characteristic supports the writeNoResponse operation.
writeDefault	boolean	Whether this characteristic supports the writeDefault operation.

©2013-2025 Tencent Cloud International Pte. Ltd.

Error Code

Error code	Error Message	Description
0	ok	Normal
-1	already connect	Connected
10000	not init	Bluetooth Adapter Not Initialized
10001	not available	The current Bluetooth adapter is unavailable.
10002	no device	Specified Device Not Found
10003	connection fail	Connection failed.
10004	no service	Specified Service Not Found
10005	no characteristic	Specified characteristic not found
10006	no connection	The current connection has been disconnected.
10007	property not support	The current characteristic does not support this operation.
10008	system error	Exceptions Reported by All Other Systems
10009	system not support	Exclusive to Android system, and BLE is not supported on versions earlier than 4.3.
10012	operate time out	Connection timeout
10013	invalid_data	The connection deviceId is either null or incorrectly formatted.

Sample Code

```
wx.getBLEDeviceCharacteristics({
  // The deviceId here needs to have already established a link with the correspond
  deviceId,
  // The ServiceId here must be obtained from the wx.getBLEDeviceServices interface
  serviceId,
  success (res) {
    console.log('device getBLEDeviceCharacteristics:', res.characteristics)
  }
})
```

createBLEConnection

This API is used via wx.createBLEConnection(Object object).

Note:

Please ensure to pair the calls to wx.createBLEConnection and wx.closeBLEConnection interfaces as much as possible. On Android, if wx.createBLEConnection is repeatedly called to create a connection, it may cause the system to hold multiple connection instances of the same device, resulting in the inability to truly disconnect from the device when calling closeBLEConnection.

The Bluetooth connection may disconnect at any time. It is recommended to monitor wx.onBLEConnectionStateChange callback events and perform reconnection operations as needed when the Bluetooth device is disconnected.

If data read and write operation interfaces are called on unconnected devices or devices that have disconnected, an error coded 10006 will be returned. It is recommended to perform reconnection operations.

Feature Description: Connects to a low-energy Bluetooth device. If the mini program has previously searched for a certain Bluetooth device and successfully established a connection, you can directly pass in the deviceId obtained from the previous search to attempt to connect to the device, eliminating the need for another search operation.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
deviceId	string	-	Yes	Bluetooth Device ID
timeout	number	-	No	Timeout period, in milliseconds (ms). If left blank, it indicates that there will be no timeout.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error code	Error Message	Description
0	ok	Normal
-1	already connect	Connected
10000	not init	Bluetooth Adapter Not Initialized

10001	not available	The current Bluetooth adapter is unavailable.
10002	no device	Specified Device Not Found
10003	connection fail	Connection failed.
10004	no service	Specified Service Not Found
10005	no characteristic	Specified characteristic not found
10006	no connection	The current connection has been disconnected.
10007	property not support	The current characteristic does not support this operation.
10008	system error	Exceptions Reported by All Other Systems
10009	system not support	Exclusive to Android system, and BLE is not supported on versions earlier than 4.3.
10012	operate time out	Connection timeout
10013	invalid_data	The connection deviceId is either null or incorrectly formatted.

Sample Code

```
wx.createBLEConnection({
  deviceId,
  success (res) {
    console.log(res)
  }
})
```

closeBLEConnection

This API is used wx.closeBLEConnection(Object object).

Feature Description: Disconnects from the low-energy Bluetooth device.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
deviceId	string	-	Yes	Bluetooth Device ID

success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error code	Error Message	Description
0	ok	Normal
-1	already connect	Connected
10000	not init	Bluetooth Adapter Not Initialized
10001	not available	The current Bluetooth adapter is unavailable.
10002	no device	Specified Device Not Found
10003	connection fail	Connection failed.
10004	no service	Specified Service Not Found
10005	no characteristic	Specified characteristic not found
10006	no connection	The current connection has been disconnected.
10007	property not support	The current characteristic does not support this operation.
10008	system error	Exceptions Reported by All Other Systems
10009	system not support	Exclusive to Android system, and BLE is not supported on versions earlier than 4.3.
10012	operate time out	Connection timeout
10013	invalid_data	The connection deviceId is either null or incorrectly formatted.

Sample Code

```
wx.closeBLEConnection({
  deviceId,
  success (res) {
    console.log(res)
```

```
    }
})
```

Bluetooth - Low Energy Peripheral Device

Last updated：2024-11-21 18:48:12

onBLEPeripheralConnectionStateChanged

This API is used via wx.onBLEPeripheralConnectionStateChanged(function listener).

Feature Description: Monitors the current peripheral device for connection or disconnection events.
Parameter and Description: function listener, the listener for when the peripheral is connected or disconnected, with the following arguments Object res:

Attribute	Type	Description
deviceId	String	Device ID with changing connection status
serverId	String	Server's UUID
connected	Boolean	Current Connection Status

offBLEPeripheralConnectionStateChanged

This API is used via wx.offBLEPeripheralConnectionStateChanged(function listener).

Feature Description: Removes the listener function for connection or disconnection events of the current peripheral device.
Parameter and Description: function listener, the listener function passed in by onBLEPeripheralConnectionStateChanged. If this parameter is not passed in, all listener functions will be removed.
Sample Code

```
const listener = function (res) { console.log(res) }

wx.onBLEPeripheralConnectionStateChanged(listener)
wx.offBLEPeripheralConnectionStateChanged(listener) // The same function object as
```

©2013-2025 Tencent Cloud International Pte. Ltd.

createBLEPeripheralServer

This API is used via wx.createBLEPeripheralServer(Object object).

Feature Description: Establishes a local server as a Bluetooth Low Energy peripheral device and allows for the creation of multiple servers.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
server	BLEPeripheralServer	Server of a peripheral device

BLEPeripheralServer

Note:

Server of a peripheral device.

.addService

This method is used via BLEPeripheralServer.addService(Object object).

Feature Description: Adds service.

Parameter and Description: Object.

Attribute	Type	Required	Description

service	Object	Yes	Object describing the service
success	function	No	Callback Function of Successful Interface Call
fail	function	No	Callback Function of Failing Interface Call
complete	function	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Structure attribute of service

Structure attributes	Type	Required	Description
uuid	string	Yes	Bluetooth service UUID
characteristics	Array.<Object>	Yes	List of Characteristics

Structure attributes of characteristics

Structure attributes	Type	Required	Description
uuid	string	Yes	UUID of characteristic
properties	Object	No	Operations Supported by the Characteristic
permission	Object	No	Permission of characteristic
value	Array.<Object>	No	Binary data corresponding to the characteristic
descriptors	Array.<Object>	No	File descriptor data

Structure attributes of properties

Structure attributes	Type	Default value	Required	Description
write	boolean	false	No	Write
writeNoResponse	boolean	false	No	Write Without Response
read	boolean	false	No	Read
notify	boolean	false	No	Subscribe
indicate	boolean	false	No	Return packet

Structure attributes of permission

Structure attributes	Type	Default value	Required	Description
readable	boolean	false	No	Readable
writeable	boolean	false	No	Writable
readEncryptionRequired	boolean	false	No	Encrypted Read Request
writeEncryptionRequired	boolean	false	No	Encrypted Write Request

Structure attributes of descriptors

Structure attributes	Type	Required	Description
uuid	string	Yes	UUID of Descriptors
permission	Object	No	Permissions of descriptors
value	ArrayBuffer	No	File descriptor data

Structure attributes of permission

Structure attributes	Type	Default value	Required	Description
write	boolean	false	No	Write
read	boolean	false	No	Read

.removeService

This method is used via BLEPeripheralServer.removeService(Object object).

Feature Description: Removes service.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
serviceId	String	-	Yes	UUID of service

success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.startAdvertising

This method is used via BLEPeripheralServer.startAdvertising(Object Object).

Feature Description: Initiates broadcasting of locally created peripheral devices.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
advertiseRequest	object	-	Yes	//Custom parameters of broadcasting
powerLevel	string	medium	No	Broadcasting power, whose valid values are: Low: Low power Medium: Moderate power High: High power
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Structure attributes of advertiseRequest

Structure attributes	Type	Default value	Required	Description
connectable	boolean	true	No	Current device's connectivity status
deviceName	string	-	No	The deviceName field in the broadcasting, which is empty by default.

serviceUuids	Array.<string>	-	No	List of service UUIDs to be broadcast. Follow the notes when using 16/32 bit UUIDs.
manufactureData	Array.<Object>	-	No	Broadcasting manufacturer information. Customization is supported on Android only and not possible on iOS due to system limitations.
beacon	object	-	No	Parameters broadcast in the form of a beacon device.

Structure attributes of manufactureData

Structure attributes	Type	Required	Description
manufacturerId	String	Yes	Manufacturer ID, a hexadecimal beginning with 0x.
manufacturerSpecificData	ArrayBuffer	No	Manufacturer information

Structure attributes of "beacon"

Structure attributes	Type	Required	Description
uuid	number	Yes	UUID broadcast by a beacon device.
major	number	Yes	Primary ID of beacon device
minor	number	Yes	Secondary ID of beacon device
measurePower	number	Yes	Reference value for determining the RSSI size when the distance to the device is 1 meter.

.stopAdvertising

This method is used via BLEPeripheralServer.stopAdvertising().

Feature Description: Ceases broadcasting.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description

success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.writeCharacteristicValue

This method is used via BLEPeripheralServer.writeCharacteristicValue(Object Object).

Feature Description: Writes binary data values into the specified characteristic and notify the connected host. As the replica node's characteristic value has changed, this interface will handle whether to take the return packet or the subscription route.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
serviceId	string	-	Yes	UUID corresponding to the Bluetooth characteristic service
characteristicId	string	-	Yes	Bluetooth characteristic UUID
value	ArrayBuffer	-	Yes	Binary data corresponding to the characteristic
needNotify	boolean	-	Yes	Whether it is necessary to notify the host that the value has been updated.
callbackId	number	-	No	Optional, used when handling return packets.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

.onCharacteristicWriteRequest

This method is used via BLEPeripheralServer.onCharacteristicWriteRequest(function listener).

Feature Description: Monitors events where the connected device requests to write the characteristic value of the current peripheral device. Upon receiving this message, it is imperative to immediately call writeCharacteristicValue to write back the data. Otherwise, the host will not receive a response.
Parameter and Description: function listener, listener for a connected device requesting to write a feature value event for the current peripheral device, with the arguments Object res as follows.

Attribute	Type	Description
serviceId	String	UUID corresponding to the Bluetooth characteristic service
characteristicId	String	Bluetooth characteristic UUID
callbackId	Number	Unique identifier, used when invoking writeCharacteristicValue.
value	ArrayBuffer	Binary data value requested for characteristic write.

.offCharacteristicWriteRequest

This method is used via BLEPeripheralServer.offCharacteristicWriteRequest(function listener).

Feature Description: Removes the listener function for events associated with characteristic value changes of the currently connected peripheral device.
Parameter and Description: function listener, the listener function passed in by onCharacteristicWriteRequest. If this parameter is not passed in, all listener functions will be removed.
Sample Code

```
const listener = function (res) { console.log(res) }

BLEPeripheralServer.onCharacteristicWriteRequest(listener)
BLEPeripheralServer.offCharacteristicWriteRequest(listener) // You need to pass the
```

.onCharacteristicReadRequest

This method is used via BLEPeripheralServer.onCharacteristicReadRequest(function listener).

Feature Description: Monitors events where the connected device requests to read the characteristic value of the current peripheral device. Upon receiving this message, it is imperative to immediately call writeCharacteristicValue to write back the data. Otherwise, the host will not receive a response.
Parameter and Description: function listener, listener for the event that a connected device requests to read the feature value of the current peripheral device, with the arguments Object res as follows.

Attribute	Type	Description
serviceId	String	UUID corresponding to the Bluetooth characteristic service
characteristicId	String	Bluetooth characteristic UUID
callbackId	Number	Unique identifier, used when invoking writeCharacteristicValue.

.offCharacteristicReadRequest

This method is used via BLEPeripheralServer.offCharacteristicReadRequest(function listener).

Feature Description: Removes the listener function for events where the connected device requests to read the characteristic value of the current peripheral device.
Parameter and Description: function listener, the listener function passed in by onCharacteristicReadRequest. If this parameter is not passed in, all listener functions will be removed.
Sample Code

```
const listener = function (res) { console.log(res) }

BLEPeripheralServer.onCharacteristicReadRequest(listener)
BLEPeripheralServer.offCharacteristicReadRequest(listener) // The same function obj
```

.onCharacteristicSubscribed

This method is used via BLEPeripheralServer.onCharacteristicSubscribed(function listener).

Feature Description: Monitors characteristic subscription events, supported exclusively on iOS.
Parameter and Description: function listener, a listener function for feature subscription events, with the following arguments Object res:

Attribute	Type	Description

serviceId	String	UUID corresponding to the Bluetooth characteristic service
characteristicId	String	Bluetooth characteristic UUID

.offCharacteristicSubscribed

This method is used via BLEPeripheralServer.offCharacteristicSubscribed(function listener).

Feature Description: Removes the listener function for characteristic subscription events.
Parameter and Description: function listener, the listener function passed in by onCharacteristicSubscribed. If this parameter is not passed in, all listener functions will be removed.
Sample Code

```
const listener = function (res) { console.log(res) }

BLEPeripheralServer.onCharacteristicSubscribed(listener)
BLEPeripheralServer.offCharacteristicSubscribed(listener) // The same function obje
```

.onCharacteristicUnsubscribed

This method is used via BLEPeripheralServer.onCharacteristicUnsubscribed(function listener).

Feature Description: Monitors characteristic unsubscription events, supported exclusively on iOS.
Parameter and Description: function listener, a listener function that cancels a feature subscription event, with the following arguments Object res:

Attribute	Type	Description
serviceId	String	UUID corresponding to the Bluetooth characteristic service
characteristicId	String	Bluetooth characteristic UUID

.offCharacteristicUnsubscribed

This method is used via BLEPeripheralServer.offCharacteristicUnsubscribed(function listener).

©2013-2025 Tencent Cloud International Pte. Ltd.

Feature Description: Removes the listener function for characteristic unsubscription events.

Parameter and Description: function listener, the listener function passed in by onCharacteristicUnsubscribed. If this parameter is not passed in, all listener functions will be removed.

Sample Code

```
const listener = function (res) { console.log(res) }

BLEPeripheralServer.onCharacteristicUnsubscribed(listener)
BLEPeripheralServer.offCharacteristicUnsubscribed(listener) // The same function ob
```

Bluetooth - Beacon

Last updated：2024-11-21 18:48:11

stopBeaconDiscovery

The method of utilizing this API is wx.stopBeaconDiscovery(Object object).

Function Description: Cease the search for nearby Beacon devices.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
0	ok	Normal
11000	unsupport	System or device not supported
11001	bluetooth service unavailable	Bluetooth service is unavailable
11002	location service unavailable	Location services are unavailable
11003	already start	Search has commenced
11004	not startBeaconDiscovery	Search has not yet commenced
11005	system error	System error
11006	invalid data	Parameters are incorrect

©2013-2025 Tencent Cloud International Pte. Ltd.

startBeaconDiscovery

The method for using this API is wx.startBeaconDiscovery(Object object)

Function Description: Initiates the search for nearby Beacon devices.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
uuids	Array.<string>	-	Supported	List of UUIDs broadcasted by Beacon devices
ignoreBluetoothAvailable	boolean	false	Not required	Verifies the Bluetooth switch, effective only on iOS. From iOS 11 onwards, even if Bluetooth is turned off in the control panel, the Beacon service can still be used.
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error Codes	Error Message	Note
0	ok	Normal
11000	unsupport	System or device not supported
11001	bluetooth service unavailable	Bluetooth service is unavailable
11002	location service unavailable	Location services are unavailable
11003	already start	Search has commenced
11004	not startBeaconDiscovery	Search has not yet commenced

©2013-2025 Tencent Cloud International Pte. Ltd.

11005	system error	System error
11006	invalid data	Parameters are incorrect

Example Code

```
wx.startBeaconDiscovery({
  success(res) { }
})
```

onBeaconUpdate

The method for using this API is wx.onBeaconUpdate(function listener)

Function Description: Monitors the Beacon device update event, only one listener can be registered.
Parameters and Description: function listener, the listener function for the Beacon device update event, with the following arguments Object res:

Attributes	Types	Note
beacons	Array.<BeaconInfo>	The list of all Beacon devices currently detected.

Example Code

```
wx.onBeaconUpdate(res => {
   console.log(res.beacons)
})
```

offBeaconUpdate

The method for using this API is wx.offBeaconUpdate()

Function Description: Removes all listener functions for the Beacon device update event.
Sample Code

```
wx.offBeaconUpdate()
```

onBeaconServiceChange

The method for using this API is wx.onBeaconServiceChange(function listener)

Function Description: Monitors the Beacon service status change event, only one listener can be registered.
Parameters and Description: function listener, listener for Beacon service state change events, with the following arguments Object res:

Attributes	Types	Note
available	boolean	Is the service currently available?
discovering	boolean	Is it currently in search mode?

Example Code

```
wx.onBeaconServiceChange(res => {
   console.log(res.available, res.discovering)
})
```

offBeaconServiceChange

The method for using this API is wx.offBeaconServiceChange()

Function Description: Removes all listener functions for the Beacon service status change event.
Sample Code

```
wx.offBeaconServiceChange()
```

getBeacons

The method for using this API is wx.getBeacons(Object object)

Function Description: Retrieves all detected Beacon devices.

Parameters and Description: Object object.

Attributes	Types	Default value	Required	Note
success	function	-	Not required	Callback function for successful interface invocation
fail	function	-	Not required	Callback function for unsuccessful interface invocation
complete	function	-	Not required	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

object.success Callback Function Parameters: Object res.

Attributes	Types	Note
beacons	Array.<BeaconInfo>	Beacon Device List

Error Code

Error Codes	Error Message	Note
0	ok	Normal
11000	unsupport	System or device not supported
11001	bluetooth service unavailable	Bluetooth service is unavailable
11002	location service unavailable	Location services are unavailable
11003	already start	Search has commenced
11004	not startBeaconDiscovery	Search has not yet commenced
11005	system error	System error
11006	invalid data	Parameters are incorrect

©2013-2025 Tencent Cloud International Pte. Ltd.

BeaconInfo

Function Description: Beacon Device.

Properties and Descriptions

Attribute	Types	Note
uuid	string	UUID broadcasted by the beacon device.
major	number	Primary ID of the beacon device.
minor	number	Secondary ID of the beacon device.
proximity	number	Enumeration value representing the device's distance (iOS only)
accuracy	number	Distance to the beacon device, measured in meters. On iOS, when proximity is 0, accuracy is -1.
rssi	number	Indicates the strength of the device's signal, measured in dBm.

Valid values for proximity

Value	Note
0	The signal is too weak to calculate the distance, or the device is not an iOS device.
1	Extremely close
2	Relatively close
3	Distant

©2013-2025 Tencent Cloud International Pte. Ltd.

Bluetooth - General

Last updated：2025-03-20 18:01:57

stopBluetoothDevicesDiscovery

This API is used via wx.stopBluetoothDevicesDiscovery(Object object).

Feature Description: Stops searching for nearby Bluetooth peripheral devices. It is advisable to invoke this interface to cease the Bluetooth search when the required Bluetooth device has been located and there is no need for further search.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error code	Error Message	Description
0	ok	Normal
-1	already connect	Connected
10000	not init	Bluetooth Adapter Not Initialized
10001	not available	The current Bluetooth adapter is unavailable.
10002	no device	Specified Device Not Found
10003	connection fail	Connection failed.
10004	no service	Specified Service Not Found

10005	no characteristic	Specified characteristic not found
10006	no connection	The current connection has been disconnected.
10007	property not support	The current characteristic does not support this operation.
10008	system error	Exceptions Reported by All Other Systems
10009	system not support	Exclusive to Android system, and BLE is not supported on versions earlier than 4.3.
10012	operate time out	Connection timeout
10013	invalid_data	The connection deviceId is either null or incorrectly formatted.

Sample Code

```
wx.stopBluetoothDevicesDiscovery({
  success (res) {
    console.log(res)
  }
})
```

startBluetoothDevicesDiscovery

This API is used via wx.startBluetoothDevicesDiscovery(Object object).

Note:

Given that Bluetooth functionality can indirectly facilitate location tracking, Android 6.0 and later versions cannot perform device searches without location permissions or if the location switch is not activated. Under these circumstances, for versions prior to Android 8.0.16, the interface call will succeed but the device cannot be scanned; for Android 8.0.16 and later versions, an error prompt will be returned.

Feature Description: Starts the search for nearby Bluetooth peripheral devices. This operation is resource-intensive, hence it is advisable to promptly call wx.stopBluetoothDevicesDiscovery to stop the search once the required device has been found.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
services	Array.		No	The list of UUIDs for the primary services of the

	<string>	-		Bluetooth devices to be searched (supporting 16/32/128 bit UUIDs). Some Bluetooth devices will broadcast their primary service's UUID. If this parameter is set, only Bluetooth devices broadcasting a primary service with the corresponding UUID will be searched. It is recommended to use this parameter to filter out other surrounding Bluetooth devices that do not need to be processed.
allowDuplicatesKey	boolean	false	No	If or not allow to report the same device more than once. If repeat reporting is allowed, the wx.onBluetoothDeviceFound method will report the same device multiple times, but with different RSSI values.
interval	number	0	No	Interval for reporting devices, in ms. 0 means that new devices are found and reported immediately, other values are reported according to the incoming interval.
powerLevel	string	medium	No	The scanning mode. The higher the mode, the faster the scan and the greater the power consumption. Only Android host clients in V 7.0.12 and later versions support this, with valid values being: Low: Low power Medium: Moderate power High: High power
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Code

Error code	Error Message	Description
0	ok	Normal
-1	already connect	Connected
10000	not init	Bluetooth Adapter Not Initialized

10001	not available	The current Bluetooth adapter is unavailable.
10002	no device	Specified Device Not Found
10003	connection fail	Connection failed.
10004	no service	Specified Service Not Found
10005	no characteristic	Specified characteristic not found
10006	no connection	The current connection has been disconnected.
10007	property not support	The current characteristic does not support this operation.
10008	system error	Exceptions Reported by All Other Systems
10009	system not support	Exclusive to Android system, and BLE is not supported on versions earlier than 4.3.
10012	operate time out	Connection timeout
10013	invalid_data	The connection deviceId is either null or incorrectly formatted.

Sample Code

```
// Taking the Bluetooth smart light of the host client's hardware platform as an ex
wx.startBluetoothDevicesDiscovery({
  services: ['FEE7'],
  success (res) {
    console.log(res)
  }
})
```

openBluetoothAdapter

This API is used via wx.openBluetoothAdapter(Object object).

Note:

Other Bluetooth-related APIs must be used after calling wx.openBluetoothAdapter. Otherwise, the API will return an error prompt (errCode=10000).

In situations where the user's Bluetooth switch is not turned on or the mobile phone does not support Bluetooth functionality, calling wx.openBluetoothAdapter will return an error code (errCode=10001), indicating that the mobile phone's Bluetooth functionality is unavailable.

Feature Description: Initializes the Bluetooth module. On iOS, when turning on the host/replica (peripheral) mode, it needs to be called once respectively, and the corresponding `mode` should be specified.

Parameter and Description: Object.

Attribute	Type	Valid Values and Descriptions	Default value	Required	Description
mode	string	central: Host mode peripheral: Replica (peripheral device) mode	central	No	Bluetooth mode, which can function as a host/replica device, only required for iOS.
success	function	-	-	No	Callback Function of Successful Interface Call
fail	function	-	-	No	Callback Function of Failing Interface Call
complete	function	-	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Return Value:

Error code	Error Message	Description
0	ok	Normal
-1	already connect	Connected
10000	not init	Bluetooth Adapter Not Initialized
10001	not available	The current Bluetooth adapter is unavailable.
10002	no device	Specified Device Not Found
10003	connection fail	Connection failed.
10004	no service	Specified Service Not Found
10005	no characteristic	Specified characteristic not found
10006	no connection	The current connection has been disconnected.

©2013-2025 Tencent Cloud International Pte. Ltd.

10007	property not support	The current characteristic does not support this operation.
10008	system error	Exceptions Reported by All Other Systems
10009	system not support	Exclusive to Android system, and BLE is not supported on versions earlier than 4.3.
10012	operate time out	Connection timeout
10013	invalid_data	The connection deviceId is either null or incorrectly formatted.

state parameter returned by the object.fail callback function (iOS only)

Status code	Description
0	Unknown
1	Resetting
2	Not supported
3	Not authorized
4	Not enabled

Sample Code

```
wx.openBluetoothAdapter({
  success (res) {
    console.log(res)
  }
})
```

getConnectedBluetoothDevices

This API is used via wx.getConnectedBluetoothDevices(Object object).

Feature Description: Gets connected Bluetooth devices based on the primary service UUID.

Parameter and Description: Object.

Attribute	Type	Default	Required	Description

		value		
services	Array.	-	Yes	List of primary service UUIDs for the Bluetooth device (supporting 16/32/128 bit UUIDs)
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Object.success callback function: Object res.

Attribute	Type	Description
devices	Array.	List of located devices

Structure attributes	Type	Description
name	string	Bluetooth device name, which may not be available for some devices.
deviceId	string	Identifier used to distinguish devices

Error Return Value

Error code	Error Message	Description
0	ok	Normal
-1	already connect	Connected
10000	not init	Bluetooth Adapter Not Initialized
10001	not available	The current Bluetooth adapter is unavailable.
10002	no device	Specified Device Not Found
10003	connection fail	Connection failed.
10004	no service	Specified Service Not Found
10005	no characteristic	Specified characteristic not found

10006	no connection	The current connection has been disconnected.
10007	property not support	The current characteristic does not support this operation.
10008	system error	Exceptions Reported by All Other Systems
10009	system not support	Exclusive to Android system, and BLE is not supported on versions earlier than 4.3.
10012	operate time out	Connection timeout
10013	invalid_data	The connection deviceId is either null or incorrectly formatted.

Sample Code

```
wx.getConnectedBluetoothDevices({
  services: ['FEE7'],
  success (res) {
    console.log(res)
  }
})
```

getBluetoothDevices

This API is used via wx.getBluetoothDevices(Object object).

Note：

The device list obtained by this interface is all the searched Bluetooth devices during the effective period of the Bluetooth module, if you do not call wx.closeBluetoothAdapter to release the resources in time after the end of the Bluetooth module use process, there will be a call to this interface will return to the previous Bluetooth device searched in the Bluetooth use process, and the device may no longer be around the user and can not be connected.

Feature Description: Gets all Bluetooth devices detected during the effective period of the Bluetooth module, including devices already connected to this device.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call

complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Object.success callback function: Object res.

Attribute	Type	Description
devices	Array.<Object>	List of devices connected corresponding to the UUID

Structure attributes	Type	Description
name	string	Bluetooth device name, which may not be available for some devices.
deviceId	string	Bluetooth Device ID
RSSI	number	Signal strength of the current Bluetooth device, measured in dBm.
advertisData	ArrayBuffer	ManufacturerData segment in the broadcast data segment of the current Bluetooth device
advertisServiceUUIDs	Array.<string>	ServiceUUIDs data segment in the broadcast data segment of the current Bluetooth device
localName	string	LocalName data segment in the broadcast data segment of the current Bluetooth device
serviceData	Object	ServiceData segment in the broadcast data segment of the current Bluetooth device
connectable	boolean	Connectivity status of the current Bluetooth device (not supported on Android 8.0 and earlier Android versions)

Sample Code

```
// ArrayBuffer to 16 progress string example
function ab2hex(buffer) {
  var hexArr = Array.prototype.map.call(
    new Uint8Array(buffer),
    function(bit) {
      return ('00' + bit.toString(16)).slice(-2)
    }
  )
  return hexArr.join('');
```

```
}
wx.getBluetoothDevices({
  success: function (res) {
    console.log(res)
    if (res.devices[0]) {
      console.log(ab2hex(res.devices[0].advertisData))
    }
  }
})
```

getBluetoothAdapterState

This API is used via wx.getBluetoothAdapterState(Object object).

Feature Description: Gets the status of the local Bluetooth adapter.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Object.success callback function: Object res.

Attribute	Type	Description
discovering	boolean	Is search for device in progress?
available	boolean	Whether the Bluetooth adapter is available.

Error Return Value

Error code	Error Message	Description
0	ok	Normal

-1	already connect	Connected
10000	not init	Bluetooth Adapter Not Initialized
10001	not available	The current Bluetooth adapter is unavailable.
10002	no device	Specified Device Not Found
10003	connection fail	Connection failed.
10004	no service	Specified Service Not Found
10005	no characteristic	Specified characteristic not found
10006	no connection	The current connection has been disconnected.
10007	property not support	The current characteristic does not support this operation.
10008	system error	Exceptions Reported by All Other Systems
10009	system not support	Exclusive to Android system, and BLE is not supported on versions earlier than 4.3.
10012	operate time out	Connection timeout
10013	invalid_data	The connection deviceId is either null or incorrectly formatted.

Sample Code

```
wx.getBluetoothAdapterState({
  success (res) {
    console.log(res)
  }
})
```

closeBluetoothAdapter

This API is used via wx.closeBluetoothAdapter(Object object).

Feature Description: Disables the Bluetooth module. Invoking this method will disconnect all established connections and free up system resources. It is recommended to pair this with wx.openBluetoothAdapter after the

Bluetooth process.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Error Return Value

Error code	Error Message	Description
0	ok	Normal
-1	already connect	Connected
10000	not init	Bluetooth Adapter Not Initialized
10001	not available	The current Bluetooth adapter is unavailable.
10002	no device	Specified Device Not Found
10003	connection fail	Connection failed.
10004	no service	Specified Service Not Found
10005	no characteristic	Specified characteristic not found
10006	no connection	The current connection has been disconnected.
10007	property not support	The current characteristic does not support this operation.
10008	system error	Exceptions Reported by All Other Systems
10009	system not support	Exclusive to Android system, and BLE is not supported on versions earlier than 4.3.
10012	operate time out	Connection timeout
10013	invalid_data	The connection deviceId is either null or incorrectly formatted.

©2013-2025 Tencent Cloud International Pte. Ltd.

Sample Code

```
wx.closeBluetoothAdapter({
  success (res) {
    console.log(res)
  }
})
```

onBluetoothDeviceFound

This API is used via wx.onBluetoothDeviceFound(function listener).

Note:

If a device is called back in wx.onBluetoothDeviceFound, this device will be added to the array obtained by the wx.getBluetoothDevices interface.

When a Bluetooth device is discovered, the system generally returns the device name from the LocalName field in the broadcast packet as the name field. However, if a connection is established with the Bluetooth device, the system's name field will change to the GattName obtained from the Bluetooth device. If there is a need to dynamically change and display the device name, it is recommended to use the localName field.

Please note that some Android models require location permissions to search for devices. Ensure that location permissions are enabled.

Feature Description: Monitors events of finding new devices.

Parameter and Description: function listener, the listener function for events that search for new devices, with the arguments object res as follows:

Attribute	Type	Description
devices	Array.<Object>	List of newly located devices

Structure attribute of devices

Structure attributes	Type	Description
name	string	Bluetooth device name, which may not be available for some devices.
deviceId	string	Bluetooth Device ID
RSSI	number	Signal strength of the current Bluetooth device, measured in dBm.

advertisData	ArrayBuffer	ManufacturerData segment in the broadcast data segment of the current Bluetooth device
advertisServiceUUIDs	Array.<string>	ServiceUUIDs data segment in the broadcast data segment of the current Bluetooth device
localName	string	LocalName data segment in the broadcast data segment of the current Bluetooth device
serviceData	Object	ServiceData segment in the broadcast data segment of the current Bluetooth device
connectable	boolean	Connectivity status of the current Bluetooth device (not supported on Android 8.0 and earlier Android versions)

Sample Code

```
// Example of converting ArrayBuffer to hexadecimal string
function ab2hex(buffer) {
  var hexArr = Array.prototype.map.call(
    new Uint8Array(buffer),
    function(bit) {
      return ('00' + bit.toString(16)).slice(-2)
    }
  )
  return hexArr.join('');
}
wx.onBluetoothDeviceFound(function(res) {
  var devices = res.devices;
  console.log('new device list has founded')
  console.dir(devices)
  console.log(ab2hex(devices[0].advertisData))
})
```

offBluetoothDeviceFound

This API is used via wx.offBluetoothDeviceFound()

Feature Description: Removes all listener functions for events of finding new devices.

Sample Code

```
wx.offBluetoothDeviceFound()
```

onBluetoothAdapterStateChange

This API is used via wx.onBluetoothAdapterStateChange(function listener).

Feature Description: Monitors events of Bluetooth adapter status changes.
Parameters and Description: Object res parameter, function listener, the listener function for events of changes to Bluetooth adapter status.

Attribute	Type	Description
available	boolean	Whether the Bluetooth adapter is available.
discovering	boolean	Whether the Bluetooth adapter is in search mode.

Sample Code

```
wx.onBluetoothAdapterStateChange(function (res) {
  console.log('adapterState changed, now is', res)
})
```

offBluetoothAdapterStateChange

This API is used via wx.offBluetoothAdapterStateChange().

Feature Description: Removes all listener functions for events of changes to Bluetooth adapter status.
Sample Code

```
wx.offBluetoothAdapterStateChange()
```

makeBluetoothPair

This API is used via wx.makeBluetoothPair(Object object).

Feature Description: Bluetooth pairing interface, only supported by Android. Normally (when you need to specify a pin code or password) the system will take over the pairing process and call wx.createBLEConnection directly. This interface should only be used if the developer doesn't want the user to enter the pin code manually and the real phone verification confirms that it works.

Parameter and Description: Object

Attribute	Type	Default value	Required	Description
deviceId	string	-	Yes	Bluetooth Device ID
pin	string	-	Yes	Pin code, in Base64 format.
timeout	number	20000	No	Timeout duration, measured in milliseconds (ms).
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

isBluetoothDevicePaired

This API is used via wx.isBluetoothDevicePaired(Object object).

Feature Description: This function queries whether a Bluetooth device is paired, and is only supported on Android.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
deviceId	string	-	Yes	Bluetooth Device ID
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful

				invocations)

©2013-2025 Tencent Cloud International Pte. Ltd.

Location

Last updated：2024-11-21 18:41:53

getLocation

This method is used via wx.getLocation(Object object).

Note:

If you want to use this interface, you need to declare it in app.json, otherwise you won't be able to use it properly.

User authorisation scope.userLocation is required before calling.

Feature Description: Get the current geolocation, speed. This interface cannot be called when the user leaves the mini program. High frequency calls can lead to power consumption, if necessary use the continuous location interface wx.onLocationChange.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
type	string	wgs84	No	WGS84 returns GPS coordinates, while GCJ02 provides coordinates that can be used with wx.openLocation.
altitude	string	false	No	Passing in "true" will return altitude information. As acquiring altitude requires high precision, it will slow down the response of the interface.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
latitude	number	Latitude, ranging from -90 to 90, with negative values indicating southern latitude.

longitude	number	Longitude, ranging from -180 to 180, with negative values indicating western longitude.
speed	number	Velocity, measured in meters per second (m/s)
accuracy	number	Precision of the location
altitude	number	Height, measured in (m)
verticalAccuracy	number	Vertical accuracy, measured in meters (m). (Cannot be obtained on Android, returning 0)
horizontalAccuracy	number	Horizontal accuracy, measured in meters (m)

Sample Code

```
wx.getLocation({
  type: 'gcj02',
  success (res) {
    const latitude = res.latitude
    const longitude = res.longitude
    const speed = res.speed
    const accuracy = res.accuracy
  }
})
```

Note:

The location simulation in the tool uses IP positioning, which may introduce some degree of error. Currently, the tool only supports gcj02 coordinates.

When using third-party services for reverse address resolution, make sure the default coordinate system of the third-party service is confirmed for accurate coordinate conversion.

choosePoi

This API is used via wx.choosePoi(Object object).

Note：

If you want to use this interface, you need to declare scope.userLocation in app.json, otherwise you won't be able to use it properly.

Feature Description: Opens the POI list to select a location, which supports both vague positioning (accurate to the city) and precise positioning in a mixed selection.

©2013-2025 Tencent Cloud International Pte. Ltd.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
type	number	When selecting a city, the value is 1; when selecting an exact location, the value is 2.
city	number	City Name
name	string	Location Name
address	string	Exact Address
latitude	number	Latitude, a floating-point number, ranging from -90 to 90, where a negative number indicates southern latitude, using the gcj02 National Administration of Surveying, Mapping and Geoinformation coordinate system (to be deprecated).
longitude	number	Longitude, a floating-point number, ranging from -180 to 180, where a negative number indicates western longitude, using the gcj02 National Administration of Surveying, Mapping and Geoinformation coordinate system (to be deprecated).

chooseLocation

Note：

If you want to use this interface, you need to declare scope.userLocation in app.json, otherwise you won't be able to use it properly.

This API is used via wx.chooseLocation(Object object).

Feature Description: Opens the map to select a location.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
type	string	tencent	No	Tencent Maps (tencent), Google Maps (google), supported by IDE only.
latitude	number	-	No	Target location's latitude.
longitude	number	-	No	Target location's longitude.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
name	string	Location Name
address	string	Exact Address
latitude	number	Latitude, a floating-point number, ranging from -90 to 90, where a negative number indicates southern latitude, using the gcj02 National Administration of Surveying, Mapping and Geoinformation coordinate system.
longitude	number	Longitude, a floating-point number, ranging from -180 to 180, where a negative number indicates western longitude, using the gcj02 National Administration of Surveying, Mapping and Geoinformation coordinate system.

stopLocationUpdate

This API is used via wx.stopLocationUpdate(Object object).

Feature Description: Stops monitoring real-time location changes, halting message reception in both the frontend and backend.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

startLocationUpdateBackground

This API is used via wx.startLocationUpdateBackground(Object object).

Feature Description: Enables the mini program to receive location messages when entering the frontend and backend, which requires guiding the user to give "authorization". Once authorized, the mini program can receive location message updates during operation or when in the backend.
Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
type	string	gcj02	No	WGS84 returns GPS coordinates, while GCJ02 provides coordinates that can be used with wx.openLocation.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

startLocationUpdate

This API is used via wx.startLocationUpdate(Object object).

Feature Description: Enables the mini program to receive location messages upon entering the frontend.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
type	string	Coordinates, gcj02	No	WGS84 returns GPS coordinates, while GCJ02 provides coordinates that can be used with wx.openLocation.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

openLocation

This API is used via wx.openLocation(Object object).

Feature Description: Uses the built-in map to view the location.

Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
type	string	tencent	No	Tencent Maps (tencent), Google Maps (google), supported by IDE only.
latitude	number	-	Yes	Latitude, ranging from -90 to 90, where a negative number indicates southern latitude, using the gcj02 National Administration of Surveying, Mapping and Geoinformation coordinate system.
longitude	number	-	Yes	Longitude, ranging from -180 to 180, where a negative number indicates western longitude, using the gcj02 National Administration of Surveying, Mapping and Geoinformation coordinate system.
scale	number	18	No	Zoom scale, ranging from 5 to 18.

©2013-2025 Tencent Cloud International Pte. Ltd.

name	string	-	No	Location Name
address	string	-	No	Detailed description of the address.
success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Sample Code

```
wx.getLocation({
 type: 'gcj02', // Returns longitude and latitude that can be used for wx.openLocat
 success (res) {
   const latitude = res.latitude
   const longitude = res.longitude
   wx.openLocation({
     latitude,
     longitude,
     scale: 18
   })
 }
})
```

onLocationChangeError

This API is used via wx.onLocationChangeError(function listener).

Feature Description: Cancels monitoring, which is triggered when the continuous location interface returns a failure.

Parameter and Description: Callback function of "function callback".

Attribute	Type	Description
errCode	number	errCode

onLocationChange

This API is used via wx.onLocationChange(function listener).

Feature Description: Monitors real-time geographical location change events, which needs to be used in conjunction with wx.startLocationUpdateBackground and wx.startLocationUpdate.

Parameter and Description: function callback, real-time geolocation change event listener function, with the following arguments Object res:

Attribute	Type	Description
latitude	number	Latitude, ranging from -90 to 90, where a negative number indicates southern latitude, using the gcj02 National Administration of Surveying, Mapping and Geoinformation coordinate system.
longitude	number	Longitude, ranging from -180 to 180, where a negative number indicates western longitude, using the gcj02 National Administration of Surveying, Mapping and Geoinformation coordinate system.
speed	number	Velocity, measured in meters per second (m/s)
accuracy	number	Precision of the location
altitude	number	Height, measured in (m)
verticalAccuracy	number	Vertical accuracy, measured in meters (m). (Cannot be obtained on Android, returning 0)
horizontalAccuracy	number	Horizontal accuracy, measured in meters (m)

Sample Code

```
const _locationChangeFn = function(res) {
 console.log('location change', res)
}
wx.onLocationChange(_locationChangeFn)
wx.offLocationChange(_locationChangeFn)
```

offLocationChangeError

This API is used via wx.offLocationChangeError(function listener).

Feature Description: Cancels monitoring, which is triggered when the continuous location interface returns a failure.
Parameter and Description: function listener, onLocationChangeError The listener function passed in. Not passing this parameter removes all listener functions.
Sample Code

```
const listener = function (res) { console.log(res) }

wx.onLocationChangeError(listener)
wx.offLocationChangeError(listener) // You need to pass the same function object as
```

offLocationChange

This API is used via wx.offLocationChange(function listener).

Feature Description: Cancels the monitoring of real-time geographical location change events.
Parameter and Description: function listener, the listener passed in by onLocationChange. Not passing this parameter removes all listener functions.
Sample Code

```
const listener = function (res) { console.log(res) }

wx.onLocationChangeError(listener)
wx.offLocationChangeError(listener) // You need to pass the same function object as
```

getFuzzyLocation

This API is used via wx.getFuzzyLocation(Object object).

Feature Description: Gets the current fuzzy geographical location.
Parameter and Description: Object.

Attribute	Type	Default value	Required	Description
type	string	wgs84	No	WGS84 returns GPS coordinates, while GCJ02 provides coordinates that can be used with wx.openLocation.

success	function	-	No	Callback Function of Successful Interface Call
fail	function	-	No	Callback Function of Failing Interface Call
complete	function	-	No	Callback function executed upon the completion of the interface invocation (both successful and unsuccessful invocations)

Parameters for object.success callback function: Object res.

Attribute	Type	Description
latitude	number	Latitude, ranging from -90 to 90, with negative values indicating southern latitude.
longitude	number	Longitude, ranging from -180 to 180, with negative values indicating western longitude.

Sample Code

```
wx.getFuzzyLocation({
 type: 'wgs84',
 success (res) {
   const latitude = res.latitude
   const longitude = res.longitude
 }
})
```

©2013-2025 Tencent Cloud International Pte. Ltd.

Open APIs

Last updated：2025-04-22 18:11:25

Login

login

The API for login is wx.login(Object object).

Feature description: This API is used to obtain a login credential. The credential can then be used to retrieve user login status information.

Notes:

The v2 version (Only supported on SaaS) can refer to the Practical Tutorial for guidance. It can be directly invoked in the IDE (minimum supported version: 2.1.111).

The v1 version requires coordination with the host client. The content returned on the actual device is provided by the host client. The host's return values can be customized. In the IDE, you can use the mock panel to mock the return values.

For version control rules, please refer to the apiVersion configuration.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls
fail	function	-	False	Callback function for failed API calls
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure)

object.success callback function parameter: Object res

Property	Type	Description
code	string	User login credential (valid for five minutes). Developers need to call jscode2sessionin the server backend, and use the code to retrieve openid and session_key.

object.fail callback function parameter: Object err

Property	Type	Description
errMsg	string	Error message.
errno	Number	An error code. For details of error codes, see Error Codes.

Example:

```
wx.login({
  success (res) {
    if (res.code) {
      // Initiate a network request
      wx.request({
        url: 'https://example.com/onLogin',
        data: {
          code: res.code
        }
      })
    } else {
      console.log('Login failed!' + res.errMsg)
    }
  }
})
```

checkSession

The API for checking session status is wx.checkSession(Object object).

Feature description: This API checks whether the login session has expired.

Notes:

1. The v2 version (Only supported on SaaS) can refer to the practice documentation for guidance. It can be directly invoked in the IDE (minimum supported version: 2.1.111).

The session_key is unique. At any given time, a user can have only one valid session_key while using the mini program.

The user login status obtained through the wx.login has a certain validity period. Developers can call the wx.checkSession to check if the current user login status is still valid.

wx.checkSession checks the session_key corresponding to the last code obtained. A successful call indicates that the session_key has not expired, while a failed call indicates that the session_key has expired.

If the login status has expired, developers can call wx.login again to obtain a new user login status.

©2013-2025 Tencent Cloud International Pte. Ltd.

2. The v1 version requires coordination with the host client. The content returned on the actual device is provided by the host client. The host's return values can be customized. In the IDE, you can use the mock panel to mock the return values.
3. For version control rules, please refer to the apiVersion configuration.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls
fail	function	-	False	Callback function for failed API calls
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure)

Example:

```
wx.checkSession({
  success () {
    // The session_key has not expired and remains valid for this lifecycle.
  },
  fail () {
    // The session_key has expired, and the login process needs to be executed agai
    wx.login() // Log in again.
  }
})
```

Account information

Notes:

APIs related to account information are supported in mini programs but not in mini games.

getAccountInfoSync

The API for getting account information is Object wx.getAccountInfoSync().

Feature description: This API retrieves current account information. The online mini program version number can only be obtained in the official version, not the development or trial versions.

Return value: Object, account information.

Property	Type	Description
miniProgram	Object	Mini program account information

miniProgram structure parameters

Structural property	Type	Description
appId	string	Mini program appId
envVersion	string	Mini program version. Valid values: develop: The development version trial: The Preview release: The official version
version	string	Online mini program version number

User information

Notes:

APIs related to account information are supported in mini programs but not in mini games.

getUserProfile

> The API for getting user profile is wx.getUserProfile(Object object).

Feature description: This API is currently not supported in IDE and needs to be jointly tested with the host client. The content return value on the device is provided by the host client, and can be customized. In the IDE, you can mock the return value on the mock panel.

Retrieves the user information. It can only be called after a page click event (for example, in the callback of bindtap on a button). An authorization popup window will appear for each request, and userInfo will be returned upon user consent. This API replaces wx.getUserInfo.

Return value: Object object.

Property	Type	Default value	Required	Description
lang	string	en	False	The language used to display the user information. Valid values: en: English zh_CN: Simplified Chinese

				zh_TW: Traditional Chinese
desc	string	-	True	Describes the purpose of retrieving user information. Up to 30 characters.
success	function	-	False	Callback function for successful API calls
fail	function	-	False	Callback function for failed API calls
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure)

getUserInfo

The API for getting user information is wx.getUserInfo(Object object).

Feature description: This API is currently not supported in IDE and needs to be jointly tested with the host client. The content return value on the device is provided by the host client, and can be customized. In the IDE, you can mock the return value on the mock panel.

To retrieve the user information, the user authorization for scope.userInfo. is required.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
lang	string	en	False	The language used to display the user information. Valid values: en: English zh_CN: Simplified Chinese zh_TW: Traditional Chinese
success	function	-	False	Callback function for successful API calls
fail	function	-	False	Callback function for failed API calls
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure)

Callback function parameter of object.success: Object res

Property	Type	Description
userInfo	UserInfo	User information object

Example

Example for the mini program user information component (user authorization)

```
<!-- If only displaying user avatar and nickname, the <open-data /> component can b
<open-data type="userAvatarUrl"></open-data>
<open-data type="userNickName"></open-data>
<!-- The button must be used for login authorization -->
<button wx:if="{{canIUse}}" open-type="getUserInfo" bindgetuserinfo="bindGetUserInf
<view wx:else>Please upgrade the host client version.</view>
```

```
Page({
  data: {
    canIUse: wx.canIUse('button.open-type.getUserInfo')
  },
  onLoad: function() {
    // Check if the user has already authorized.
    wx.getSetting({
      success (res){
        if (res.authSetting['scope.userInfo']) {
          // Already authorized, can directly call getUserInfo to get user profile
          wx.getUserInfo({
            success: function(res) {
              console.log(res.userInfo)
            }
          })
        }
      }
    })
  },
  bindGetUserInfo (e) {
    console.log(e.detail.userInfo)
  }
})
```

Example:

```
// Must be called when the user has already authorized.
wx.getUserInfo({
  success: function(res) {
    var userInfo = res.userInfo
    var nickName = userInfo.nickName
    var avatarUrl = userInfo.avatarUrl
  }
})
```

userInfo

Feature description: User information.

Parameter and description:

Property	Type	Description
nickName	string	User’s nickname.
avatarUrl	string	The URL of the user's profile photo. The last number in the URL represents the size of the square profile photo (options are 0, 46, 64, 96, 132; 0 represents a 640x640 square profile photo, 46 represents a 46x46 square profile photo, and so on. The default value is 132.) This field is left blank if the user has no profile photo. If the user changes the profile photo, the original profile photo URL will become invalid.
gender	number	User's gender. No longer returned. Valid values are: 0: Unknown 1: Male 2: Female
country	string	The user’s country. No longer returned.
province	string	The user's province. No longer returned.
city	string	The user's city. No longer returned.
language	string	The language used to display the country, province, and city. Valid values: en: English zh_CN: Simplified Chinese zh_TW: Traditional Chinese

Settings

AuthSetting

Feature description: User’s authorization settings information.

Parameter and description:

Property	Description
boolean scope.userLocation	Whether to authorize location access, corresponding to the wx.getLocation and wx.chooseLocation
boolean scope.writePhotosAlbum	Whether to authorize saving to the photo album, corresponding to the wx.saveImageToPhotosAlbum.

boolean scope.camera	Whether to authorize camera access, corresponding to the <camera /> component.
boolean scope.addFriend	Allows to be added as friends. Call wx.authorize for authorization.

getSetting

The API for settings is wx.getSetting(Object object).

Feature description: Gets the current settings of the user.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls
fail	function	-	False	Callback function for failed API calls
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure)

object.success callback function parameter: Object res.

Property	Type	Description
authSetting	AuthSetting	User authorization result.

Example:

```
wx.getSetting({
  success(res) {
    console.log(res.authSetting)
    // res.authSetting = {
    // "scope.userInfo": true,
    // "scope.userLocation": true
    // }
  }
})
```

openSetting

The API for opening settings page is wx.openSetting(Object object).

Feature description: This API opens the mini program settings page in the client, returning the user's settings operation results. After the user taps, they can navigate to the settings page to manage authorization information.
Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls
fail	function	-	False	Callback function for failed API calls
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure)

object.success callback function parameter: Object res.

Property	Type	Description
authSetting	AuthSetting	User authorization result.

Example:

```
wx.openSetting({
  success(res) {
    console.log(res.authSetting)
    // res.authSetting = {
    // "scope.userInfo": true,
    // "scope.userLocation": true
    // }
  }
})
```

```
wx.openSetting({
  success (res) {
    console.log(res.authSetting)
    // res.authSetting = {
    // "scope.userInfo": true,
    // "scope.userLocation": true
    // }
  }
})
```

Biometric authentication

Notes:

APIs related to biometric authentication are supported in mini programs but not in mini games.

checkIsSoterEnrolledInDevice

The API for checking biometric information is wx.checkIsSoterEnrolledInDevice(Object object).

Feature description: This API checks whether biometric information is entered into the device.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls
fail	function	-	False	Callback function for failed API calls
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure)

object.success callback function parameter: Object res.

Property	Type	Description
isEnrolled	boolean	Whether the information has been entered.
errMsg	string	Error message.

Example:

```
wx.checkIsSoterEnrolledInDevice({
  success(res) {
    console.log(res.isEnrolled)
  }
})
```

checkIsSupportSoterAuthentication

The API for checking biometric authentication support is wx.checkIsSupportSoterAuthentication(Object object).

Feature description: This API checks whether the current device supports biometric authentication. If supported, the success callback is triggered; if not supported, the fail callback is triggered.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
success	function	-	False	Callback function for successful API calls
fail	function	-	False	Callback function for failed API calls
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure)

Example:

```
wx.checkIsSupportSoterAuthentication({
  success() {
    // Support biometric authentication
   }
})
```

startSoterAuthentication

The API for starting biometric authentication is wx.startSoterAuthentication(Object object).

Feature description: This API starts the biometric authentication.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
authContent	string	''	False	Verification description, i.e. the prompts displayed during the authentication
success	function	-	False	Callback function for successful API calls
fail	function	-	False	Callback function for failed API calls

complete	function	-	False	Callback function executed after API call ends (regardless of success or failure)

Example:

```
wx.startSoterAuthentication({
   authContent: 'Unlock biometric authentication',
   success() {
     // Authentication successful
   }
})
```

Authorization

authorize

The API for authorization is wx.authorize(Object object).

Feature description: This API initiates an authorization request to the user in advance. After this API is called, a window will pop up, asking the user whether they agree to authorize the mini program to use a certain feature or access certain user data. However, the corresponding APIs are not actually called. If the user has previously agreed to the authorization, no pop-up window will appear and it will return success directly.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
scope	string	-	True	Scope of the required permissions. For more information, see Scope list.
success	function	-	False	Callback function for successful API calls
fail	function	-	False	Callback function for failed API calls
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure)

Example:

```
// You can query whether the user has authorized the scope of "scope.record" using
```

```
wx.getSetting({
  success(res) {
    if (!res.authSetting['scope.record']) {
      wx.authorize({
        scope: 'scope.record',
        success() {
          // Once the user has granted permission for the mini program to use the r
          wx.startRecord()
        }
      })
    }
  }
})
```

Subscription

requestSubscribeMessage

The API for subscription is wx.requestSubscribeMessage(Object object).

Feature description: This API triggers the client mini program's subscription message page and returns the user's subscription message operation result. You can learn the user's subscription status for related template messages using wx.getSetting .

Notes:

Only supported on SaaS.

One-time template IDs and permanent template IDs cannot be used simultaneously.

In a single authorization call, each tmplId must correspond to a unique template title. If there are duplicates, only one will be retained.

Parameter and description: Object object.

Property	Type	Default value	Required	Description
tmplIds	Array	-	True	Array of message template IDs to subscribe to. A maximum of 3 messages can be subscribed to in one call. Template IDs are configured in the console under Mini program management - Subscription. Each tmplId must correspond to a unique template title, otherwise duplicates will be filtered out.

success	function	-	False	Callback function for successful API calls
fail	function	-	False	Callback function for failed API calls
complete	function	-	False	Callback function executed after API call ends (regardless of success or failure)

object.success callback function parameter: Object res

Property	Type	Description
errMsg	String	The value of errMsg is 'requestSubscribeMessage:ok' when the API is called successfully.
[TEMPLATE_ID: string]	String	[TEMPLATE_ID] is a dynamic key representing the template ID. The value can be 'accept', 'reject', 'ban', or 'filter'. Accept: The user agreed to subscribe to the message corresponding to this template ID. Reject: The user refused to subscribe to the message corresponding to this template ID. Ban: The template has been banned by the backend. Filter: The template was filtered out by the backend due to duplicate titles. For example, { errMsg: "requestSubscribeMessage:ok", zun-LzcQyW-edafCVvzPkK4de2Rllr1fFpw2A_x0oXE: "accept"} indicates that the user has agreed to subscribe to the message zun-LzcQyW-edafCVvzPkK4de2Rllr1fFpw2A_x0oXE.

object.fail callback function parameter: Object err

Property	Type	Description
errMsg	string	Error message when the API call fails.
errno	Number	Error code when the API call fails.

Error code

errCode	errMsg	Description
10001	TmplIds can't be empty.	The tmplIds parameter is empty.
10002	Failed to request the	Network error. Failed to request the message list.

	message list.	
10003	Failed to send the subscription request.	Network error. Failed to send the subscription request.
20001	No template data return, verify if the template ID exists.	No template data, usually because the template ID does not exist or does not match the template type.
20002	Templates type must be the same.	Both one-time and permanent template IDs are used.
20003	There are too many template messages.	The number of template messages exceeds the limit.
20004	The main switch is switched off.	Subscriptions cannot be made because the user has switched off the main switch.
20005	This mini program is banned from subscribing messages.	The mini program is banned from subscribing messages.
20013	Reject DeviceMsg Template	Subscription to device messages using this API is not allowed.

Example:

```
wx.requestSubscribeMessage({
  tmplIds: [''],
  success: (res) => {
    console.log('requestSubscribeMessage===success', res)
```

```
    }
})
```

Custom APIs

Last updated：2025-01-16 19:28:13

invokeNativePlugin

Our mini program SDK offers interfaces that enable the implementation of certain Open Platform APIs, which can be directly transmitted to the host client, such as login and user information retrieval. Some host clients can be expanded to maintain UI and functionality consistency, such as scanning and sharing. Additionally, it is possible to extend custom APIs for direct transmission to the host client.

Usage method at mini program end

```
var opts = {
  api_name: '', // Name of the API
  success: function(res) {},
  fail: function(res) {},
  complete: function(res) {},
  data: { // Input Parameters
     name : 'kka',
     age : 22,
     data: {...}
  }
}
wx.invokeNativePlugin(opts); // Invocation
```

Methods

Method	Description
View onCreateLoadingView()	Create a loading view, permitting the host client to render mini program/mini game brand and custom display styles. If returned as null, the default loading view will be exhibited.
View onCreateCapsuleView()	Establish a capsule button, allowing the host to customize the capsule button style. If returned as null, the default capsule button will be displayed.
boolean onShowMenu()	Triggered when the "..." more button is clicked. If onCreateCapsuleView returns null and onShowMenu returns false, the default menu will be displayed.

void onExit()	Activated when the "◯" exit button is clicked. If onCreateCapsuleView returns null and onExit returns false, the default operation will be executed.
boolean onAuthorize(JSONObject params, Value Callback callback)	Implementation of the wx.authorize interface. If it returns false, the notification of wx api call failure will be directly sent.
boolean onOpenSetting(JSONObject params, ValueCallback callback)	Implementation of the wx.openSetting interface. If it returns false, the notification of wx api call failure will be directly sent.
boolean onGetSetting(JSONObject params, ValueCallback callback)	Implementation of the wx.getSetting interface. If it returns false, the notification of wx api call failure will be directly sent.
boolean onLogin(JSONObject params, ValueCallback callback)	Implementation of the wx.login interface. If it returns false, the notification of wx api call failure will be directly sent.
boolean onRefreshSession(JSONObject params, ValueCallback callback)	Implementation of the wx.checkSession interface. If it returns false, the notification of wx api call failure will be directly sent.
boolean onRequestPayment(JSONObject params, ValueCallback callback)	Implementation of the wx.requestPayment interface. If it returns false, the notification of wx api call failure will be directly sent.
boolean onGetUserInfo(JSONObject params, ValueCallback callback)	Implementation of the wx.getUserInfo interface. If it returns false, the notification of wx api call failure will be directly sent. For more details, see onGetUserInfo.
boolean onShareAppMessage(JSONObject params, ValueCallback callback)	Implementation of the wx.shareAppMessage interface. If it returns false, the notification of wx api call failure will be directly sent.
boolean onNavigateToMiniProgram(JSONObject params, ValueCallback callback)	Implementation of the wx.navigateToMiniProgram interface. If it returns false, the notification of wx api call failure will be directly sent.
boolean onScanCode(JSONObject params, ValueCallback callback)	Implementation of the wx.scanCode interface. If it returns false, the notification of wx api call failure will be directly sent.
boolean onOpenDocument(JSONObject params, ValueCallback callback)	Implementation of the wx.openDocument interface. If it returns false, the notification of wx api call failure will be directly sent.
boolean onOpenLocation(JSONObject params, ValueCallback callback)	Implementation of the wx.openLocation interface. If it returns false, the notification of wx api call failure will be directly sent.
boolean onChooseLocation(JSONObject params, ValueCallback callback)	Implementation of the wx.chooseLocation interface. If it returns false, the notification of wx api call failure will be directly sent.
boolean onPreviewImage(JSONObject	Implementation of the wx.rviewImage interface. If it returns

params, ValueCallback callback)	false, the notification of wx api call failure will be directly sent.
boolean onChooseImage(JSONObject params, ValueCallback callback)	Implementation of the wx.chooseImage interface. If it returns false, the notification of wx api call failure will be directly sent.
boolean onChooseVideo(JSONObject params, ValueCallback callback)	Implementation of the wx.chooseVideo interface. If it returns false, the notification of wx api call failure will be directly sent.
boolean onShowToast(JSONObject params, ValueCallback callback)	Implementation of the wx.showToast and wx.showLoading interfaces. If it returns false, the default style will be displayed.
boolean onHideToast(JSONObject params, ValueCallback callback)	Implementation of the wx.hideToast and wx.hideLoading interfaces. If it returns false, the default style will be displayed.
boolean onShowModal(JSONObject params, ValueCallback callback)	Implementation of the wx.showModal interface. If it returns false, the default style will be displayed.
boolean onInvokeWebAPI(String event, JSONObject params, ValueCallback callback)	Implementation of the custom wx api. If it returns false, the notification of wx api call failure will be directly sent. For more details, see onInvokeWebAPI.
boolean onReportEvent(String event, Map params)	For more details, see onReportEvent.

boolean onGetUserInfo(JSONObject params, ValueCallback callback)

```
-(void)onGetUserInfoWithParams:(NSDictionary *)params inApp:(NSString *)appId callb
    //TODO
        NSDictionary* userInfo = @{
                                   @"nickname" : @"morven",
                                   @"headimgurl" : @"https://wx.qlogo.cn/mmopen/vi_
                                   @"sex" : @(1),
                                   @"province" : @"Guangdong",
                                   @"city" : @"Shenzhen",
                                   @"country" : @"China"
                                   };
        NSMutableDictionary* result = [NSMutableDictionary dictionaryWithCapacity:1
        NSMutableDictionary* userInfoDic = [NSMutableDictionary dictionaryWithCapac
        NSMutableDictionary* resultDataDic = [NSMutableDictionary dictionaryWithCap
        [userInfoDic setValue:userInfo[@"nickname"] forKey:@"nickName"];
        [userInfoDic setValue:userInfo[@"headimgurl"] forKey:@"avatarUrl"];
        [userInfoDic setValue:userInfo[@"sex"] forKey:@"gender"];
        [userInfoDic setValue:userInfo[@"province"] forKey:@"province"];
        [userInfoDic setValue:userInfo[@"city"] forKey:@"city"];
        [userInfoDic setValue:userInfo[@"country"] forKey:@"country"];

        NSData *data=[NSJSONSerialization dataWithJSONObject:userInfoDic options:NS
```

```
        [resultDataDic setValue:[[NSString alloc]initWithData:data encoding:NSUTF8S
        [result setValue:resultDataDic forKey:@"data"];
        [result setValue:@"getUserInfo:ok" forKey:@"errMsg"];

    if (handler) {
        //success
        handler(result, nil);
    }
}
```

boolean onInvokeWebAPI(String event, JSONObject params, ValueCallback callback)

```
- (BOOL)onInvokeWebAPIWithEvent:(NSString*)event params:(NSDictionary*)params callb
    NSLog(@"onInvokeWebAPIWithEvent, event:%@, params:%@", event, params);
    if (handler) {
        handler(@{@"errMsg" : @"onInvokeWebAPIWithEvent"}, nil);
    }
    return YES;
}
```

boolean onReportEvent(String event, Map<String, String> params)

```
Event reporting. The events are as follows:
 1. Launch the mini program, event: MS_EVENT_LAUNCH, keys for params: pagePath,
d;
 2. Duration of mini program usage, event: MS_EVENT_USE_TIME, keys for params:
useTime, startId, appId; the unit of useTime is milliseconds;
 3. Successful launch of the mini program, event: MS_EVENT_LAUNCH_SUCCESS, keys
for params: appId.
 4. Opening a page within the mini program, event: MS_EVENT_OPEN_PAGE, keys for
params: pagePath, appId.
 5. Failure to launch the mini program, event: MS_EVENT_LAUNCH_FAIL, keys for
params: pagePath, reason, appId.
```

Sensitive APIs

Last updated：2025-02-24 17:10:36

Warning:

The APIs listed on this page are sensitive and involve accessing user information. Developers are advised to read the relevant API documentation below and submit applications for approval as needed. If any illegal or non-compliant behavior is detected, the platform reserves the right to revoke access, suspend accounts, and pursue legal action.

Location

getLocation

Feature: Gets the current geographical location and speed.

Description: Gets the current geographical location and speed. For details, see getLocation.

choosePoi

Note:

Only supported in mini programs.

Feature: Opens a POI (Point of Interest) list to select a location.

Description: Opens the POI list to select the location. Supports both fuzzy (city-level) and precise location selection. For details, see choosePoi.

chooseLocation

Note:

Only supported in mini programs.

Feature: Opens the map to select the location.

Description: Opens the map to select the location. For details, see chooseLocation.

startLocationUpdateBackground

Note:

Only supported in mini programs.

Feature: Enables the mini program to receive location messages when switching to the frontend and backend.

Description: To enable the mini program to receive location messages when switching to the frontend and backend, the user needs to grant the permission first. For details, see startLocationUpdateBackground.

startLocationUpdate

Note:

Only supported in mini programs.

Feature: Enables the mini program to receive location messages when switching to the frontend.

Description: Enables the mini program to receive location messages when switching to the frontend. For details, see startLocationUpdate.

onLocationChange

Note:

Only supported in mini programs.

Feature: Listens for real-time geographic location change event.

Description: To listen for real-time geographic location change event, this API should be used together with wx.startLocationUpdateBackground and wx.startLocationUpdate. For details, see onLocationChange.

getFuzzyLocation

Feature: Gets the current fuzzy geographic location.

Description: Gets the current fuzzy geographical location. For details, see getFuzzyLocation.

Open APIs

startSoterAuthentication

Note:

Only supported in mini programs.

Feature: Initiates biometric authentication.

Description: Initiates SOTER biometric authentication (calls system API). For details, see startSoterAuthentication.

checkIsSupportSoterAuthentication

Note:

Only supported in mini programs.

Feature: Gets the biometric authentication method.

Description: Gets the biometric authentication methods supported by the device (calls system API).

For details, see checkIsSupportSoterAuthentication.

checkIsSoterEnrolledInDevice

Note:

Only supported in mini programs.

Feature: Calls the biometric authentication API.

Description: Checks whether biometric information, such as fingerprints, has been enrolled in the device (call the system API). For details, see checkIsSoterEnrolledInDevice.

©2013-2025 Tencent Cloud International Pte. Ltd.

Contacts

chooseContact

Note:

Only supported in mini programs.

Feature: Accesses the Contacts in the phone.

Description: Accesses the Contacts in the phone and selects the desired contact. For details, see chooseContact.

H5 Real-Time Communication

Last updated：2025-03-20 18:01:56

In order to help customers quickly migrate existing H5 pages, the mini program supports embedding H5 pages within mini programs using the web-view component. Additionally, we support H5 pages to call mini-program APIs using JSSDK and support real-time communication between H5 pages and mini-programs. This is not only a technical innovation, but also a key factor in optimizing the user experience.

Data sharing: By establishing an effective communication mechanism, the user's actions and status in the mini program can be transmitted to the H5 page in real time and vice versa. This instant sharing of data ensures user consistency across pages, making it easier for users to complete tasks.
Capability expansion: The integration of mini-program with H5 pages extends the overall capabilities. For example, a mini-program can call specific functions within an H5 page or process data from an H5 page.
Improve development efficiency: Developers can leverage existing H5 page resources without redeveloping the same features. The communication mechanism enables better integration between mini programs and H5 pages, reducing migration, development, and maintenance costs.
Adapt to diverse needs: Different users may have varying habits and requirements. The communication mechanism allows developers to flexibly adjust interactions between mini programs and H5 pages based on user feedback and needs.

In summary, enabling communication between mini programs and embedded H5 pages via the `web-view` component not only enhances the user experience but also provides developers with greater flexibility and convenience.

sendWebviewEvent

Feature description: The mini program sends message events to the H5 page embedded within the `web-view` component.
Example: For more information about the real-time communication, seeCommunication section in JSSDK - API description.

The mini program sends message events to H5 page

JAVASCRIPT
H5 JAVASCRIPT

```
wx.sendWebviewEvent({
```

```
  message: "I'm Miniprogram, I received"
})
```

```
function onEvent(e) {
  console.log(e.message) // I'm Miniprogram, I received
}
// Register a listener
wx.miniProgram.onWebviewEvent(onEvent)
// Remove the listener
wx.miniProgram.offWebviewEvent(onEvent)
```

Ad

Last updated：2025-04-09 18:35:50

wx.createRewardedVideoAd

wx.createRewardedVideoAd(Object object)

Feature description: Creates a rewarded video ad component. The rewarded video ad created by this method is a singleton within a page and cannot be used across pages in the mini program.

Parameter: Object object.

Property	Type	Default value	Required	Description
adUnitId	string	-	True	Ad unit ID.
multiton	boolean	-	False	Whether to enable multiple-instance mode. Default: false.

Return value: RewardedVideoAd**.**Rewarded video ad component.

RewardedVideoAd

The rewarded video ad will be rendered as a native component with a higher level than regular components. The rewarded video ad object is a **singleton within a page**. By default, the rewarded ad is hidden and needs to be displayed using RewardedVideoAd.show().

RewardedVideoAd.load()

Feature description: Loads the rewarded video ad.

Return value:**Promise**, the result of loading the rewarded video ad data.

RewardedVideoAd.show()

Feature description: Displays a rewarded video ad.

Return value:**Promise**, the result of displaying the rewarded video ad.

RewardedVideoAd.destroy()

Feature description: Destroys a rewarded video ad instance.

RewardedVideoAd.onLoad(function listener)

Feature description: Listens for the rewarded video ad loading event.

Parameter: function listener. The listener for the rewarded video ad loading event.

©2013-2025 Tencent Cloud International Pte. Ltd.

RewardedVideoAd.offLoad(function listener)

Feature description: Removes the listener for the rewarded video ad loading event.

Parameter: function listener. The listener that was passed to onLoad. If not provided, all listeners will be removed.

Example:

```
const listener = function (res) { console.log(res) }

RewardedVideoAd.onLoad(listener)
RewardedVideoAd.offLoad(listener) // Must pass the same function object used in onL
```

RewardedVideoAd.onError(function listener)

Feature description: Listens for rewarded video error events.

Parameter: function listener. The listener for rewarded video error events.

Return parameter: Object res.

Property	Type	Description
errMsg	string	Error information.
errCode	number	Error code.

RewardedVideoAd.offError(function listener)

Feature description: Removes the listener for rewarded video error events.

Parameter: function listener. The listener that was passed to onError. If not provided, all listeners will be removed.

Example:

```
const listener = function (res) { console.log(res) }

RewardedVideoAd.onError(listener)
RewardedVideoAd.offError(listener) // Must pass the same function object used in on
```

RewardedVideoAd.onClose(function listener)

Feature description: Listens for the event when a user taps the `close` button.

Parameter: function listener. The listener for the event when a user taps the `close` button.

Return parameter: Object res.

Property	Type	Description
isEnded	boolean	Whether the video was closed after the user watched it completely.

RewardedVideoAd.offClose(function listener)

Feature description: Removes the listener for the event when a user taps the `close` button.

Parameter: function listener. The listener that was passed to onClose. If not provided, all listeners will be removed.

Example:

```
const listener = function (res) { console.log(res) }

RewardedVideoAd.onClose(listener)
RewardedVideoAd.offClose(listener) // Must pass the same function object used in on
```

Error code

Error codes are obtained through onError event. During debugging, you can capture error details via exception returns. You can add appropriate monitoring information to assist in troubleshooting issues in the live environment.

Error code	Description	Reason
1000	Failed to call the backend API	This error is not caused by the developer.
1001	Parameter error	Incorrect usage.
1002	Invalid ad unit	Incorrect ad unit ID spelling or accidental use of another app's ad unit ID.
1003	Internal error	This error is not caused by the developer.
1004	No suitable ad	Ads do not appear every time; this time it might not appear because there are no ads that match the user's preferences.
1010	Ad feature disabled	The ad feature has been disabled.

©2013-2025 Tencent Cloud International Pte. Ltd.

Server Backend
Mini Program Login

Last updated：2025-06-06 18:27:20

Note：
Only supported on SaaS.

jscode2session

Request method: Get
Request URL: `/openserver/sns/jscode2session`

Request parameters

Property	Type	Required	Description
appid	string	True	Mini program appId.
secret	string	True	Mini program appSecret.
js_code	string	True	Code obtained during login.
grant_type	string	True	Authorization type: use authorization_code

Response parameters

Property	Type	Required	Description
session_key	string	True	Session key.
errmsg	string	True	Error message.
openid	string	True	Unique user identifier.
unionid	string	True	The same user will have the same UnionID when logging into different mini-programs or mini games in the same app.
errcode	int32	True	Error code.

```
{
```

```
    "errcode": 40029,
    "errmsg": "invalid js_code",
    "openid": "",
    "requestId": "84574d84227943298cb25e26a6d2b816",
    "session_key": "",
    "unionid": ""
}
```


getAccessToken

Request method: Get

Request URL: `/openserver/cgi-bin/token`

Request parameters

Property	Type	Required	Note
appid	string	True	Mini program appId.
secret	string	True	Mini program appSecret.
grant_type	string	True	Enter client_credential.

Response parameters

Property	Type	Required	Note
access_token	string	True	Access token.
expires_in	number	True	Token validity period in seconds. Value range: usually 7200 seconds.


```
{
    "access_token": "ATZrSqJXfZkAZVRIJxTNZJwyZLZJjPJhKpibbjuGgHpQzZwbCDPvGGfWKSbJlS
    "errcode": 0,
    "errmsg": "ok",
    "expires_in": 7200,
    "requestId": "5486f23fd3b64c2c84c9798ead510aa7"
}
```


©2013-2025 Tencent Cloud International Pte. Ltd.

checkSessionKey

Request method: Get

Request URL: `/openserver/wxa/checksession`

Request parameters

Property	Type	Required	Note
access_token	string	True	Access token.
openid	string	True	Unique user identifier.
signature	string	True	User login state signature, obtained by signing an empty string with session_key. That is, signature = hmac_sha256(session_key, "").
sig_method	string	True	Hash method for user login state signature, currently only supports hmac_sha256.

Response parameters

Property	Type	Required	Note
errcode	number	True	Error code
errmsg	string	True	Error message.

getuserphonenumber

Request method:：post。

Request URL：/openserver/wxa/business/getuserphonenumber。

Request parameters

Property	Type	Required	Note
access_token	string	True	Access token.
openid	string	True	Unique user identifier.
code	string	True	temporary code

```
{
    "access_token":"ATIfWyEpUVHcyGHHqCEspuuALGNyoBFpLwTNdRMLsdJAdJPEJHvrEYSlhZQtJoj
    "openid":"b642cf744a16d2010b60242ab6b7ded99a895a60e857a9dc17eadcb73f790cfd",
    "code":"807636166d9c4155aacc7fbeae7b9bfa"
}
```

Response parameters

Property	Type	Note
errcode	number	Error code.
errmsg	string	Error message.
phone_info	object	Phone info
phone_info.phoneNumber	string	Phone mumber. This parameter is obtained by calling the JS SDK wx.getPhoneNumber API

```
{
    "errcode": 0,
    "errmsg": "ok",
    "phone_info": {
        "phoneNumber": "xxxxxx"
    }
}
```

getemailaddress

Request method:：post。

Request URL：/openserver/wxa/business/getemailaddress。

Request parameters

Property	Type	Required	Note
access_token	string	True	Access token.
openid	string	True	Unique user identifier.
code	string	True	temporary code. This parameter is obtained by calling the JS SDK wx.getEmailAddress API

```
{
    "access_token":"ATIfWyEpUVHcyGHHqCEspuuALGNyoBFpLwTNdRMLsdJAdJPEJHvrEYSlhZQtJoj
    "openid":"b642cf744a16d2010b60242ab6b7ded99a895a60e857a9dc17eadcb73f790cfd",
    "code":"807636166d9c4155aacc7fbeae7b9bfa"
}
```


Response parameters

Property	Type	Note
errcode	number	Error code.
errmsg	string	Error message.
email_address	object	email_address
email_address.emailAddress	string	emailAddress


```
{
    "errcode": 0,
    "errmsg": "ok",
    "email_address": {
        "emailAddress": "xxxxxx"
    }
}
```


©2013-2025 Tencent Cloud International Pte. Ltd.

Subscription Message

Last updated：2025-01-17 11:04:26

Note：

Only supported on SaaS.

Query personal template list

Request method: Get

Request URL: `/openserver/wxaapi/newtmpl/gettemplate`

Request parameters

Property	Type	Required	Description
access_token	string	True	Access token

Response parameters

Property	Type	Description
errmsg	string	Error message.
errcode	int32	Error code.
data	array<object>	Template list.
data.priTmplId	string	Template ID.
data.title	string	Title.
data.content	string	Content.
data.example	string	Example.
data.type	string	Type.
data.keywordEnumValueList	array<object>	Enumeration list.
data.keywordEnumValueList.keywordCode	string	Enumeration name.

©2013-2025 Tencent Cloud International Pte. Ltd.

data.keywordEnumValueList.enumValueList	array<string>	Enumeration values.


```
{
    "data": [
        {
            "content": "testtest400:{{phrase1.DATA}}\\ntesttest401:{{phrase2.DATA}}
            "example": "testtest400:testtest400\\ntesttest401:testtest401",
            "keywordEnumValueList": [
                {
                    "enumValueList": [
                        "To be paid",
                        "Wait for delivery",
                        "Has been shipped",
                        "Completed",
                        "Canceled"
                    ],
                    "keywordCode": "enum.DATA"
                }
            ],
            "priTmplId": "mti_oAWECeEacUIdYvUxSbJBNbkOIfzJHEcJIRSEMAE",
            "title": "testtest400",
            "type": 3
        },
        {
            "content": "Order Number:{{thing1.DATA}}\\nOrder type:{{thing2.DATA}}\\
            "example": "Order Number:100020241130100001\\nOrder type:Personal order
            "keywordEnumValueList": [
                {
                    "enumValueList": [
                        "To be paid",
                        "Wait for delivery",
                        "Has been shipped",
                        "Completed",
                        "Canceled"
                    ],
                    "keywordCode": "enum.DATA"
                }
            ],
            "priTmplId": "mti_AgKglmMmSMbFVXBpaFunPSIoDGddPKSFGbSAYRe",
            "title": "Received order notification",
            "type": 2
        }
    ],
    "errcode": 0,
    "errmsg": "ok",
```

```
    "requestId": "19d63f1109f24857961de213bcfc7e75"
}
```

Send subscription messages

Request method: Post

Request URL: `/openserver/cgi-bin/message/subscribe/send`

Request parameters

Property	Type.	Required	Description
access_token	string	True	Access token for the API call. This parameter should be in the URL, not in the body.
template_id	string	True	The ID of the subscription template to be sent.
page	string	False	The page to navigate to when the template card is clicked. This must be a page within the mini program. Parameters can be included (e.g. index?foo=bar). If not provided, the template will not have a navigation link.
touser	string	True	The OpenID of the recipient (user).
data	string	True	Template content.
miniprogram_state	string	True	The type of mini program to navigate to: developer for the development version, trial for the Preview, formal for the official version. Defaults to the official version.
lang	string	True	Language types viewed in the mini program. Supported values are zh_CN (Simplified Chinese) and en_US (English).

Response parameters

Property	Type.	Description
errcode	number	Error code.
errmsg	string	Error message.

```
{"errcode":0,"errmsg":"ok"}
```

Additional description

Subscription message parameter value limitations

Parameter	Category	Parameter value limitations	Regular Expression
thing.DATA	Thing	Up to 20 characters	^[a-zA-Z0-9!@#$%^&*()_+={}\\[\\]:;\\"\\'<>,.?/~`-]{1,20}$
number.DATA	Number	Up to 32 numbers	^[0-9]{1,32}(\\.[0-9]+)?$
letter.DATA	Letter	Up to 32 letters	^[a-zA-Z]{1,32}$
symbol.DATA	Symbol	Up to 5 symbols	^[^\\w\\s]{1,5}$
character_string.DATA	String	Up to 64 characters	^[a-zA-Z0-9\\W_]{1,64}$
time.DATA	Time	24-hour time format, two times connected by a "~" symbol, format: hh: mm: ss or hh: mm	^(?:(?:([01]\\d|2[0-3]):([0-5]\\d)(?::([0-5]\\d))?(?:~([01]\\d|2[0-3]):([0-5]\\d)(?::([0-5]\\d))?)?)?)?$
date.DATA	Date	In the format of year, month, and day, connect two time with a "~" symbol. such as yyyy-MM-dd hh:mm:ss or yyyy-MM-dd	^(\\d{4}-[01]\\d-[0-3]\\d(?: (?:[01]\\d|2[0-3]):[0-5]\\d:[0-5]\\d)?)(?:~(\\d{4}-[01]\\d-[0-3]\\d(?: (?:[01]\\d|2[0-3]):[0-5]\\d:[0-5]\\d)?))?$
amount.DATA	Amount	1 currency symbol plus numbers with up to 10 numbers, and decimals are supported	^[A-Za-z$€¥]{1}[\\d]{0,8}(\\.\\d{1,2})?$
phone_number.DATA	Phone number	Up to 17 characters. Numbers and symbols are allowed.	^[\\d\\-\\+$$\\s]{1,17}$
car_number.DATA	Car plate	Up to 12 characters	^.{1,12}$
name.DATA	Name	Up to 32 characters	^. {1,32}$
phrase.DATA	Description	Up to 16 characters	^.{1,16}$

For example, the content of the template could be:

```
Name: {{name01.DATA}}
Amount: {{amount01.DATA}}
Itinerary: {{thing01.DATA}}
Date: {{date01.DATA}}
```

The corresponding JSON:

```
{
    "touser": "OPENID",
    "template_id": "TEMPLATE_ID",
    "page": "index",
    "data": {
        "name01": {
            "value": "XX"
        },
        "amount01": {
            "value": "$100"
        },
        "thing01": {
            "value": "Guangzhou to Beijing"
        },
        "date01": {
            "value": "2018-01-01"
        }
    }
}
```

©2013-2025 Tencent Cloud International Pte. Ltd.

Mini Program Payment

Last updated：2025-04-10 15:18:17

1. Query order status

API description:

Queries order status using the order number.

Request method: Get

Request URL:

/v3/pay/transactions/out-trade-no/:out_trade_no

Request HEADER

Parameter	Required	Description	Note
Authorization	True	Signature authentication information	Refer to APIv3 Certificate Signature
Content-Type	True	application/json	-

Request parameters

Property	Type	Required	Description
out_trade_no	string	True	Order number.

Response parameters

Property	Type	Required	Description
appid	string	True	Mini program ID.
mch_id	string	True	Merchant account, provided when the merchant places the order.
out_trade_no	string	True	Merchant order number, provided when the merchant places the order.
transaction_id	string	False	Unique identifier of the payment order, which is returned when the order is paid successfully.

©2013-2025 Tencent Cloud International Pte. Ltd.

trade_type	string	False	Transaction type: JSAPI for mini program payments.
trade_state	string	True	Transaction status: WAIT_PAY - Awaiting payment Success - Payment successful CLOSED - Order closed AUTO_CLOSED - Order automatically closed due to expiration REFUND - Refunded PAY_ERROR - Payment failed
bank_type	string	False	Bank code. Non-bank card payment types are uniformly marked as OTHERS.
attach	string	False	Merchant data packet, custom data provided by the merchant when placing the order, not visible to the user, up to 256 characters. If provided, it will be returned as-is in this API, payment success callback notification, and transaction bill. If not provided, it will not be returned.
success_time	string	False	Payment completion time. It follows the RFC 3339 standard format: yyyy-MM-DDTHH:mm:ss+TIMEZONE. yyyy-MM-DD represents the date; T separates the date and time; HH:mm:ss represents the time; TIMEZONE represents the time zone (e.g., +08:00 for Beijing time).
payer	payer	False	Information about the payer, returned when the order is paid successfully.
amount	orderAmount	False	Order amount.

Response example:


```
{
    "appid": "mpco56h12e6e52hj",
    "mch_id": "mi_7b0a5e40f9",
    "out_trade_no": "2b695106b888d14328d9",
    "transaction_id": "pi1738915449351wbrgic93qlahw34134wjr",
    "trade_type": "JSAP",
```

```
    "trade_state": "SUCCESS",
    "bank_type": "OTHERS",
    "attach": "attch info",
    "success_time": "2025-02-28T10:34:56+08:00",
    "payer": {
        "openid": "o910d4edeee717377adguZS89513"
    },
    "amount": {
        "payer_total": "88800",
        "total": 88800,
        "currency": "USD",
        "payer_currency": "USD"
    }
}
```


2. Close an order

API description:

Closes the original order to avoid duplicate payments when a merchant's order payment fails, and a new order number is generated to re-initiate the payment.
Additionally, if the user payment times out after the system places an order and the system exits without further processing, the close order API should be called to prevent the user from continuing the payment.

Request method: Post

Request URL:

/v3/pay/transactions/out-trade-no/:out_trade_no/close

Request HEADER

Parameter	Required	Description	Note
Authorization	True	Signature authentication information	Refer to APIv3 Certificate Signature
Content-Type	True	application/json	-

Request parameters

Property	Type	Required	Description
out_trade_no	string	True	Merchant order number, provided when the merchant places the order.
mchid	string	True	Merchant account, provided when the

			merchant places the order.

Response parameters

Property	Type	Required	Description

Response example:

No response packet

```
204 No Content
```

3. Place an order in a mini program

API description:

In the mini program payment scenario, merchants need to call this API to place an order in the application and generate a prepay transaction session identifier (prepay_id) used to initiate the payment.

Request method: Post

Request URL:

/v3/pay/transactions/jsapi

Request HEADER

Parameter	Required	Description	Note
Authorization	True	Signature authentication information	Refer to APIv3 Certificate Signature
Content-Type	True	application/json	-

Request parameters

Property	Type	Required	Description
appid	string	True	Mini program ID.
description	string	True	Product description, up to 127 characters.
out_trade_no	string	True	Internal order number in the merchant system, must be 6-32 characters long, can only include numbers, uppercase and

			lowercase letters, _-|*, and must be unique within the same merchant ID.
time_expire	string	True	1. Definition: The payment end time is the last time the user can complete the payment for this order, not the order closing time. After this time, the user will not be able to pay for the order. To close an order, call the order closing API. 2. Format requirements: The payment end time shall follow the RFC 3339 standard format: yyyy-MM-DDTHH:mm:ss+TIMEZONE. yyyy-MM-DD represents the date; T separates the date and time; HH:mm:ss represents the time; TIMEZONE represents the time zone (e.g., +08:00 for Beijing time). 3. Parameters can only be configured when the user first places an order, and subsequent modifications are not allowed, attempts to modify will result in errors. If the actual payment time exceeds the set payment end time, the merchant needs to place a new order with a new merchant order number for the user to pay. If the payment end time has not been exceeded, the original parameters can be used to request the order API again to get the latest prepay_id for payment. The payment end time cannot be earlier than 1 minute after the order time. If the set payment end time is earlier, the system will automatically adjust it to 1 minute after the order time.
attach	string	False	Custom data packet provided by the merchant when creating the order, not visible to the user, used to store merchant custom information related to the order, with a total length limit of 128 characters. This field will be returned to the merchant in the order query API and payment success callback notification, and will also appear in the transaction bill.

notify_url	string	True	The URL where the merchant receives payment success callback notifications. It must comply with the notify_url specification.
amount	amount	True	Order amount.
payer	payer	True	Payer information.
detail	detail	True	Discount feature.

Response parameters

Property	Type	Required	Description
prepay_id	string	True	Prepay transaction session identifier, required for JSAPI or mini program payment initiation, valid for 2 hours. If it expires, a new prepay_id must be requested by calling this API again.

Response example:

```
{
    "prepay_id":"pi173898442419368bhz26jn4rikpwzr1akn"
}
```

4. Payment callback

API description:

When a user successfully pays for an order, the application payment system will send a callback notification to the merchant's pre-configured callback URL (notify_url) via a POST request, informing the merchant that the user has completed the payment.

Request method: Post

Request URL:

notify_url passed in by the order API

Parameter	Description	Note
Wechatpay-Serial	Serial No. of the superapp payment certificate verified	Refer to Superapp Payment Certificate Signature

Wechatpay-Signature	Value of signature to be verified	-
Wechatpay-Timestamp	Timestamp for signature verification	-
Wechatpay-Nonce	Random string for signature verification	-

Request parameters

Property	Type	Required	Description
id	string	True	Unique identifier for the callback notification.
create_time	string	True	The time when the callback notification was created. Format: Follow the RFC 3339 standard format: yyyy-MM-DDTHH:mm:ss+TIMEZONE. yyyy-MM-DD represents the year, month, and day. the “T” character is used to separate the date and time portions. HH:mm:ss represents the specific hour, minute, and second. TIMEZONE represents the time zone (for example, +08:00 corresponds to the East 8th Zone time, which is Beijing Time). Example: 2015-05-20T13:29:35+08:00 represents 13:29:35 on May 20, 2015 (Beijing time).
resource_type	string	True	Resource data type of the notification. It is fixed as encrypt-resource.
event_type	string	True	Type of payment callback notification. For successful payment notifications, the type is TRANSACTION.SUCCESS.
pay_model	string	False	Type of payment bank card, OTHERS for non-bank card payments.
summary	string	True	The summary note for callback content in the payment system, limited to 64 characters.

resource	paymentNotifyResource	True	Notification resource data.

Response parameters

Property	Type	Required	Description
ok	string	True	-

Response example:

```
{
  "ok"
}
```

Data types

paymentNotifyResource

Property	Type	Required	Description
original_type	string	True	Original callback type: The object type before encryption, which is transaction.
algorithm	string	True	Encryption algorithm type: The encryption algorithm type for the callback data ciphertext. It is currently AEAD_AES_256_GCM. Developers need to use the same type of data for decryption.
ciphertext	string	True	Data ciphertext: The Base64-encoded ciphertext of the callback data. Merchants need to Base64 decode and decrypt it.
associated_data	string	False	Additional data: Additional data involved in decryption. This field may be empty.
nonce	string	True	Random string: The random string involved in decryption.

detail

Property	Type	Required	Description
cost_price	integer	False	Original price of the order.

©2013-2025 Tencent Cloud International Pte. Ltd.

goods_detail	array[goods_detail]	False	List of individual product details.

goods_detail

Property	Type	Required	Description
merchant_goods_id	string	False	Unified product ID defined by the payment system (optional).
goods_name	string	False	Actual name of the product.
quantity	integer	True	Quantity of products purchased by the user.
unit_price	integer	True	Integer, unit: cents. If the merchant offers a discount, the discounted unit price should be provided (e.g., if a user uses a 100-50 coupon on a 100 USD order, the unit price of the promotional product should be the original unit price minus 50).

payer

Property	Type	Required	Description
openid	string	False	Unique identifier for the user under the merchant's appid.

orderAmount

Property	Type	Required	Description
payer_total	string	False	Total order amount (in cents).
total	interger	False	Actual amount paid by the user, an integer value, in cents. User payment amount = total amount - voucher amount.
currency	string	False	Currency type: CNY - Chinese Yuan, USD -

			US Dollar.
payer_currency	string	False	User payment currency: CNY - Chinese Yuan, USD - US Dollar.

amount

Property	Type	Required	Description
total	interger	False	Actual amount paid by the user, an integer value, in cents. User payment amount = total amount - voucher amount.
currency	string	False	Currency type: CNY - Chinese Yuan, USD - US Dollar.

Guidelines for setting notify_url

（1）The notify_url should be the actual address of the merchant's own system, not an example address from the API documentation or demo.

（2）The notify_url must be a complete URL starting with either https:// or http://. Ensure that the domain and IP in the URL are accessible from the internet; do not use localhost, 127.0.0.1, 192.168.x.x, or any local or internal network IPs.

（3）The notify_url should not include any parameters.

Common errors:

Error description	Error example
URL only contains a domain, missing a specific path.	http://www.weixin.qq.com
URL does not start with https:// or http://, missing domain or IP.	./PayNotify.aspx
URL contains a local or internal network IP.	http://127.0.0.1/pay/notify.php
Non-URL formatted string.	xxxxxxx, 1234567, test

Notes on callback handling logic

The backend code handling the notify_url should not include login state verification.

Upon receiving a payment result notification, the merchant system must respond within 5 seconds. Otherwise, the application payment system will consider the notification failed and will resend it.
The same notification may be sent multiple times to the merchant system. The merchant system must correctly handle duplicate notifications. If the notification has already been processed, directly return a success response to the payment system.

©2013-2025 Tencent Cloud International Pte. Ltd.

Error Code

Last updated：2025-01-09 17:07:13

Note：

Only supported on SaaS.

Error code	Description	Note
-1	System error	System error. Please try again later.
500	Database query error	Internal error. Database execution exception.
10002	Invalid request parameter	Invalid request parameter.
40001	Invalid token	AppSecret is incorrect when getting the access_token, or the access_token is invalid. Verify the AppSecret or check if the correct public account is calling the API.
40002	Invalid grant_type	Invalid authorization type
40003	Invalid OpenID	Invalid OpenID. Check if the OpenID (the current user) has followed the public account, or if it is an OpenID of another public account.
40013	Invalid app ID	Invalid app ID. Check the input parameter.
40014	Invalid access_token	Invalid access_token. Check the validity of the access_token (such as whether it is expired), or check check if the correct public account is calling the API.
40015	Invalid account ID	Incorrect account ID. Check the input parameter.
40029	Invalid code	Invalid js_code.
40037	Invalid template_id	Incorrect template ID. Check the input parameter.
40038	The number of message push exceeds the threshold	The daily limit for long-term message templates is 5 messages.
43101	User does not subscribe the message	Check the subscription popup callback results or event push to confirm if the user has successfully subscribed.
43107	Subscription message capability banned	Check if the account is banned from sending subscription messages or if the template corresponding to the template_id is banned.

©2013-2025 Tencent Cloud International Pte. Ltd.

43108	Concurrent message sending error	Check for concurrent requests.
45168	Sensitive words	Check if the delivered message contains any sensitive words.
47003	Data structure format	Verify the data structure format and ensure each keyword meets the corresponding rules.

©2013-2025 Tencent Cloud International Pte. Ltd.

JS SDK
web-view

Last updated：2025-04-29 18:40:52

When mini-program developers use the web-view component and want to call mini program API capabilities within an H5 page, they need to implement this using JSSDK.
JSSDK is a toolkit for web developers. By using the JSSDK, web developers can use some of the native capabilities (e.g. camera, album, audio, location, etc.) within the corresponding H5 page to provide users with a better web experience.
Since JSSDK is implemented based on the mini program's web-view component container, it can only be used when the link is opened within a mini-program. For domain management of the webpage, please refer to Platform management or Mini program management - Development management.

How to use

Please load the corresponding JSSDK in the web page, refer to the following code:

```
<script src="path/to/jssdk.js"></script>

<script>
    // go to pay Page from click in H5 Page
    document.getElementById("payBtn").click = funciton(e){
        wx.miniProgram.navigateTo({url: "/pages/pay/pay"});
    }
</script>
```

Online address

JSSDK 1.0.0
JSSDK 1.0.1

API Description

Last updated：2025-05-26 15:25:45

JSSDK Compatibility Checklist

JSSDK	IDE	Base Library	iOS SDK	Android SDK
1.0.0	2.0.812	2.0.8	2.0.7	2.0.7
1.0.1	2.2.611	2.2.6	2.2.6	2.2.6

Routes

API	Description	Minimum Version
wx.miniProgram.navigateTo	Retains the current page and navigates to a specific page within the app. However, it cannot navigate to a tabBar page.	1.0.0
wx.miniProgram.navigateBack	Closes the current page and returns to the previous page or a multi-level page.	1.0.0
wx.miniProgram.switchTab	Redirects to the tabBar page and closes all other non-tabBar pages.	1.0.0
wx.miniProgram.reLaunch	Closes all pages and opens a specified page in the app.	1.0.0
wx.miniProgram.redirectTo	Closes the current page and navigates to a specific page within the app. However, it cannot navigate to a tabBar page.	1.0.0

Notes:

JSSDK Routes APIs are consistent with the mini program Routes APIs.However, it cannot trigger the `success` / `fail` / `complete` callback functions.

Example:

JAVASCRIPT

```
wx.miniProgram.navigateTo({url: '/path/to/page'})
```

```
wx.miniProgram.navigateBack({delta: 1})
wx.miniProgram.switchTab({url: '/path/to/page'})
wx.miniProgram.reLaunch({url: '/path/to/page'})
wx.miniProgram.redirectTo({url: '/path/to/page'})
```

Navigation

API	Description	Minimum Version
wx.navigateToMiniProgram	Opens another mini program.	1.0.0
wx.exitMiniProgram	Closes the current mini program.	1.0.0
wx.navigateBackMiniProgram	Returns to the previous mini program. This can only be called successfully when the current mini program is opened by another mini program.	1.0.0

Notes:

JSSDK navigation APIs are consistent with the mini program navigation APIs.

Example:

JAVASCRIPT

```
wx.navigateToMiniProgram({
  appId: 'appId',
  envVersion: 'release',
  success: (res) => {
    console.log('navigateToMiniProgram success', res)
  },
  fail: (err) => {
    console.log('navigateToMiniProgram fail', err)
  }
})

wx.exitMiniProgram({
  success: (res) => {
    console.log('exitMiniProgram success', res)
  },
  fail: (err) => {
    console.log('exitMiniProgram fail', err)
  }
})
```

```
wx.navigateBackMiniProgram({
  success: (res) => {
    console.log('navigateBackMiniProgram success', res)
  },
  fail: (err) => {
    console.log('navigateBackMiniProgram fail', err)
  }
})
```

Navigation bar

API	Description	Minimum Version
wx.setNavigationBarTitle	Dynamically sets the title of the navigation bar for the current page.	1.0.0

Notes:

JSSDK navigation bar APIs are consistent with the mini program navigation bar APIs.

Example:

JAVASCRIPT

```
wx.setNavigationBarTitle({
  title: 'title',
  success: (res) => {
    console.log('setNavigationBarTitle success', res)
  },
  fail: (err) => {
    console.log('setNavigationBarTitle fail', err)
  }
})
```

Data cache

API	Description	Minimum Version
wx.getStorage	Asynchronously gets the content of the specified key from the local cache.	1.0.0

wx.setStorage	Stores the data to the specified key in the local cache, which will overwrite the original content associated with that key.	1.0.0
wx.removeStorage	Removes the specified key from the local cache.	1.0.0
wx.clearStorage	Clears the local data cache.	1.0.0
wx.getStorageInfo	Asynchronously gets the information of the current storage.	1.0.0

Notes:

JSSDK data cache APIs are consistent with the mini program cache APIs.

Example:

JAVASCRIPT

```
wx.getStorage({
  key: 'name',
  success: (res) => {
    console.log('getStorage success', res)
  },
  fail: (err) => {
    console.log('getStorage fail', err)
  }
})

wx.setStorage({
  key: 'name',
  data: 'value',
  success: (res) => {
    console.log('setStorage success', res)
  },
  fail: (err) => {
    console.log('setStorage fail', err)
  }
})

wx.removeStorage({
  key: 'name',
  success: (res) => {
    console.log('removeStorage success', res)
  },
  fail: (err) => {
    console.log('removeStorage fail', err)
  }
})

wx.clearStorage({
```

```
  success: (res) => {
    console.log('clearStorage success', res)
  },
  fail: (err) => {
    console.log('clearStorage fail', err)
  }
})

wx.getStorageInfo({
  success: (res) => {
    console.log('getStorageInfo success', res)
  },
  fail: (err) => {
    console.log('getStorageInfo fail', err)
  }
})
```

Media

Category	Description	API	Minimum Version
Image APIs	Specifies to choose an image from the local album or take a photo.	wx.chooseImage	1.0.0
	Previews an image in full screen mode on a new page.	wx.previewImage	1.0.0
	Gets the Base64 local image.	wx.getLocalImgData	iOS: 1.0.0 Android: 1.0.1
Recording APIs.	Starts recording	wx.startRecord	1.0.0
	Stops recording.	wx.stopRecord	1.0.0
Audio APIs	Starts audio playback.	wx.playVoice	1.0.0
	Pauses audio playback.	wx.pauseVoice	1.0.0
	Stops audio playback.	wx.stopVoice	1.0.0

Notes:

JSSDK media APIs are consistent with the mini program media APIs.

Example:

JAVASCRIPT

```
wx.chooseImage({
  count: 2,
  sizeType: ['original', 'compressed'],
  sourceType: ['album', 'camera'],
  success: (res) => {
    console.log('chooseImage success', res)

    const imagePaths = res.tempFilePaths
    wx.previewImage({
      urls: imagePaths,
      success: (res) => {
        console.log('previewImage success', res)
      },
      fail: (err) => {
        console.log('previewImage fail', err)
      }
    })

    wx.getLocalImgData({
      filePath: imagePaths[0],
      success: (res) => {
        console.log('getLocalImgData success', res)
      },
      fail: (err) => {
        console.log('getLocalImgData fail', err)
      }
    })
  },
  fail: (err) => {
    console.log('chooseImage fail', err)
  }
})

wx.startRecord({
  success: (res) => {
    console.log('startRecord success', res)

    const recordFilePath = res.tempFilePath
    wx.playVoice({
      filePath: recordFilePath,
      success: (res) => {
        console.log('playVoice success', res)
      },
      fail: (err) => {
```

```
        console.log('playVoice fail', err)
      }
    })
  },
  fail: (err) => {
    console.log('startRecord fail', err)
  }
})

wx.stopRecord({
  success: (res) => {
    console.log('stopRecord success', res)
  },
  fail: (err) => {
    console.log('stopRecord fail', err)
  }
})

wx.pauseVoice({
  success: (res) => {
    console.log('pauseVoice success', res)
  },
  fail: (err) => {
    console.log('pauseVoice fail', err)
  }
})

wx.stopVoice({
  success: (res) => {
    console.log('stopVoice success', res)
  },
  fail: (err) => {
    console.log('stopVoice fail', err)
  }
})
```

Location

API	Description	Minimum Version
wx.getLocation	Gets current location info.	1.0.0
wx.openLocation	Views location using the app's built-in map.	1.0.0

wx.chooseLocation	Opens the map and selects a location.	1.0.0

Notes:

JSSDK location APIs are consistent with the mini program location APIs.

Example:

JAVASCRIPT

```
wx.getLocation({
  success: (res) => {
    console.log('getLocation success', res)

    const latitude = res.latitude
    const longitude = res.longitude
    wx.openLocation({
      latitude,
      longitude,
      success: (res) => {
        console.log('openLocation success', res)
      },
      fail: (err) => {
        console.log('openLocation fail', err)
      }
    })
     wx.chooseLocation({
      latitude,
      longitude,
      success: (res) => {
        console.log('chooseLocation success', res)
      },
      fail: (err) => {
        console.log('chooseLocation fail', err)
      }
    })
  fail: (err) => {
    console.log('getLocation fail', err)
  }
})
```

Document

API	Description	Minimum Version

wx.openDocument	Opens the document (supports only PDF on Android, and supports the file types opened by the system's WKWebView on iOS).	1.0.0

Notes:

JSSDK document APIs are consistent with the Mini program document APIs.

`filePath` only supports local file paths, such as those obtained using wx.downloadFile.

Example:

JAVASCRIPT

```
wx.openDocument({
  filePath: '/path/to/file',
  success: (res) => {
    console.log('openDocument success', res)
  },
  fail: (err) => {
    console.log('openDocument fail', err)
  }
})
```

Open APIs

API	Description	Minimum Version
wx.login	This API is used to obtain a login credential. The credential can then be used to retrieve user login status information.	1.0.1
wx.checkSession	This API checks whether the login session has expired.	1.0.1
wx.getPhoneNumber	Get the dynamic token code for exchanging the user's cell phone number.	1.0.1
wx.getEmailAddress	Get the dynamic token code used to exchange for the user's email address.	1.0.1
wx.chooseAvatar	Show user avatar panel and get the avatar url after user action.	1.0.1
wx.getNickname	Display the user nickname panel and get the nickName when the user clicks on the nickname panel.	1.0.1
wx.hideNicknameAccessory	Hide User Nickname Panel.	1.0.1

©2013-2025 Tencent Cloud International Pte. Ltd.

wx.requestPayment	Bring up the application payment.	1.0.1

Example:

JAVASCRIPT

```
wx.login({
  success: (res) => {
    console.log('login success', res.code)
  },
  fail: (err) => {
    console.log('login fail', err)
  }
})

wx.checkSession({
  success: () => {
    console.log('session_key has not expired')
  },
  fail: () => {
    console.log('session_key has expired, please log in again')
  }
})

wx.getPhoneNumber({
  success: (res) => {
    console.log('getPhoneNumber success', res.code)
  },
  fail: (err) => {
    console.log('getPhoneNumber fail', err)
  }
})

wx.getEmailAddress({
  success: (res) => {
    console.log('getEmailAddress success', res.code)
  },
  fail: (err) => {
    console.log('getEmailAddress fail', err)
  }
})

wx.chooseAvatar({
  success: (res) => {
    console.log('chooseAvatar success', res.avatarUrl)
    const { avatarUrl } = res;
    if (avatarUrl.includes('/tmp')) {
```

```
      wx.getLocalImgData({
        filePath: avatarUrl,
        success: (res) => {
          console.log('getLocalImgData success', res.localData)
        },
        fail: (err) => {
          console.log('getLocalImgData fail', err)
        }
      })
    }
  },
  fail: (err) => {
    console.log('chooseAvatar fail', err)
  }
})

// Recommended to be called when clicking on an input box
wx.getNickname({
  success: (res) => {
    console.log('getNickname success', res.nickName)
    // Here you can populate the input box with the nickName
  },
  fail: (err) => {
    console.log('getNickname fail', err)
  }
})

// Recommended to be called when the input box is out of focus
wx.hideNicknameAccessory({
  success: (res) => {
    console.log('hideNicknameAccessory success', res)
  },
  fail: (err) => {
    console.log('hideNicknameAccessory fail', err)
  }
})

wx.requestPayment({
  package,
  nonceStr,
  paySign,
  signType,
  timeStamp,
  success: (res) => {
    console.log('requestPayment success', res)
  },
  fail: (err) => {
```

```
    console.log('requestPayment fail', err)
  }
})
```

Device

API	Description	Minimum Version
wx.scanCode	Opens the client's image that has the QR code for scanning.	1.0.0
wx.getNetworkType	Gets the network type.	1.0.0

Notes:

JSSDK device APIs are consistent with the mini program device APIs.

Example:

JAVASCRIPT

```
wx.scanCode({
  success: (res) => {
    console.log('scanCode success', res)
  },
  fail: (err) => {
    console.log('scanCode fail', err)
  }
})

wx.getNetworkType({
  success: (res) => {
    console.log('getNetworkType success', res)
  },
  fail: (err) => {
    console.log('getNetworkType fail', err)
  }
})
```

Communication

API	Description	Minimum Version

wx.miniProgram.postMessage	Sends a message to the mini program. This will trigger the web-view component's `message` event during back navigation, component destruction, or sharing.	1.0.0
wx.miniProgram.onWebviewEvent	Registers a listener to receive message events sent from the mini program to the H5 page.	1.0.0
wx.miniProgram.offWebviewEvent	Removes the listener to stop receiving message events sent from the mini program to the H5 page.	1.0.0
wx.miniProgram.sendWebviewEvent	Sends a message to the mini program, triggering the web-view component's `event` .	1.0.0

H5 page sends a message event to the mini program

H5 JAVASCRIPT

WXML

JAVASCRIPT

```
// H5 javascript
wx.miniProgram.postMessage({ data: {name: 'hi data'} })

wx.miniProgram.sendWebviewEvent({ data: {message: 'hi data'} })
```

```
<!-- Mini program /pages/index/index.wxml -->
<web-view src="https://xxx.xxx" bindmessage="onMessage" bindevent="onEvent">
</web-view>
```

```
Page({
  onMessage(e) {
    console.log(e.detail.data) // [{"name":"hi data"}]
  },
  onEvent(e) {
    console.log(e.detail.data) // {"message":"hi data"}
  }
}
```

Mini program sends message event to H5 page

JAVASCRIPT

H5 JAVASCRIPT

```
wx.sendWebviewEvent({
  message: "I'm Miniprogram, I received"
```

```
})
```

```
function onEvent(e) {
  console.log(e.message) // I'm Miniprogram, I received
}
// Register a listener
wx.miniProgram.onWebviewEvent(onEvent)
// Remove the listener
wx.miniProgram.offWebviewEvent(onEvent)
```

Others

API	Description	Minimum Version
wx.miniProgram.getEnv	Gets the current environment.	1.0.0
wx.canGoBack	Checks if the current H5 page can navigate back to the previous page.	1.0.0
wx.checkJsApi	Determines if the current client version supports a specified JS API.	1.0.0
wx.invokeNativePlugin	Calls custom APIs within an H5 page.	1.0.0
wx.showAppMenu	Display the "More" panel of the host app, which is the same as the "···" operation in the capsule button.	1.0.1

Gets the current environment

There are three methods to determine if the `H5` page is embedded within the mini program's `web-view` component:

Use `wx.miniProgram.getEnv`

```
// H5 javascript
wx.miniProgram.getEnv((res) => {
  console.log(res.miniprogram) // true
})
```

Use `window.__wxjs_environment` Invoke it within the callback of `WeixinJSBridgeReady` event.

Example:

```
// H5 javascript
```

```
function ready() {
  console.log(window.__wxjs_environment === 'miniprogram') // true
}
if (!window.WeixinJSBridge || !WeixinJSBridge.invoke) {
  document.addEventListener('WeixinJSBridgeReady', ready, false)
} else  {
  ready()
}
```

Check if the `userAgent` string contains the keyword `miniProgram` .

Example:

```
// H5 javascript
// "Mozilla/..... miniProgram TMA/mp6iu47n4rps6zwz": Contains miniProgram and the
console.log(window.navigator.userAgent);
```

Gets the web-view URL

Gets the current `web-view` URL when users share. This can be done by returning the webViewUrl parameter in the `onShareAppMessage` call back. `webViewUrl`

Example:

```
// Mini program javascript
Page({
  onShareAppMessage(options) {
    console.log(options.webViewUrl)
  }
})
```

Determines if the H5 page can navigate back

This involves checking for a boolean value that indicates whether there is a previous page or state in the current navigation stack that can be returned to.

Example:

```
// H5 javascript
wx.canGoBack({
  success: (res) => {
    console.log('canGoBack success', res.canGoBack) // true/false
  },
  fail: (err) => {
    console.log('canGoBack fail', err)
  }
})
```

Gets client version info about the JS API support

Determines if the current client version supports a specified JS API.

Example:

```
// H5 javascript
wx.checkJsApi({
  jsApiList: ['chooseImage'],
  success: (res) => {
    // Return as a key-value pair, true for available APIs and false for unavailabl
    // For example: {"checkResult":{"chooseImage":true},"errMsg":"checkJsApi:ok"}
    console.log('checkJsApi success', res)
  },
  fail: (err) => {
    console.log('checkJsApi fail', err)
  }
})
```

Calls custom APIs

Calls custom APIs within an H5 page.

Example:

```
// H5 javascript
invokeNativePlugin() {
  let opts = {
    api_name: 'customApiName', // Custom API name
    data: JSON.parse(this.apiData || '{}'), // API request parameters
    success: (res) => {
      console.log('invokeNativePlugin success', res)
    },
    fail: (err) => {
      console.log('invokeNativePlugin fail', err)
    }
  }
  wx.invokeNativePlugin(opts); // Call custom APIs
}
```

Display the "More" panel of the superapp

Display the "More" panel of the superapp, which is the same as the "···" operation in the capsule button.

Example：

```
// H5 javascript
wx.showAppMenu({
  success: (res) => {
```

```
    console.log('showAppMenu success', res)
  },
  fail: (err) => {
    console.log('showAppMenu fail', err)
  }
})
```

Base Library
Base Library Introduction

Last updated：2025-01-16 19:31:09

Introduction

The base library acts as a middleware responsible for loading mini programs. It pre-packages many capabilities of the app and the phone (such as Bluetooth, NFC, etc.), allowing developers to use standard components and APIs to develop mini programs.
You can think of the base library as similar to common web development libraries like React or Lodash. However, it is more specialized because the capabilities of mini programs require support from the SDK. Therefore:
The base library, like the host app, has its own version number.
New capabilities added in each version of the base library need to run in specific versions.
Some capabilities of higher versions of the base library are not compatible with lower version SDKs.

Typically, the base library provides two main parts for mini program developers: components and APIs.
WXML files in mini programs, when compiled, can render real DOM nodes in the environment provided by the base library.
APIs like Page and Component used in JS files of mini programs are also exposed globally by the base library.
The wx.xxx APIs used in mini programs actually call native capabilities (like Bluetooth, WiFi, etc.), and the base library acts as a "bridge" between the mini program and the native container.

In short, the base library mainly handles data binding, provides a component system, event system, communication system, and other framework logic. From the dual-process model perspective, it includes rendering issues of the Virtual DOM in the rendering layer and built-in components and APIs in the logic layer. Additionally, the base library provides supplementary capabilities for mini programs, such as custom components and performance compatibility.

```
const {SDKVersion} = wx.getSystemInfoSync(); // fetch jsLib version
```

Base library mechanism

When developing web pages, we often reference third-party JavaScript libraries. Before using the APIs provided by these libraries, we need to include them in our business code.

Similarly, when opening a mini program in an app, we need to load the base library before starting the mini program, and then load the business code. Since the rendering layer and logic layer of mini programs are managed by two

©2013-2025 Tencent Cloud International Pte. Ltd.

threads, when we talk about the base library, it usually includes the WebView base library (rendering layer) and the AppService base library (logic layer).

Before opening any mini program based on our platform, the same base library needs to be injected. Therefore, we do not need to package the base library in the mini program's code package. Instead, you need to integrate our mini program SDK in the app and introduce the mini program base library through the SDK. Introducing the base library through the SDK has two benefits:
It can reduce the size of the mini program code package.
It allows adding capabilities or fixing bugs in the base library without modifying the mini program's code.

Mini program startup
When you open a mini program, you'll see a loading page that includes the mini program's icon and name. On this page, the base library and code package are downloaded first. Once the downloads are complete, the base library and code package are loaded separately.

©2013-2025 Tencent Cloud International Pte. Ltd.

Base Library Maintenance

Last updated：2025-01-16 19:33:10

Base library FAQs

1. **Who can update the base library version?**

Our official platform.

2. **How long does it take for the base library update to take effect?**

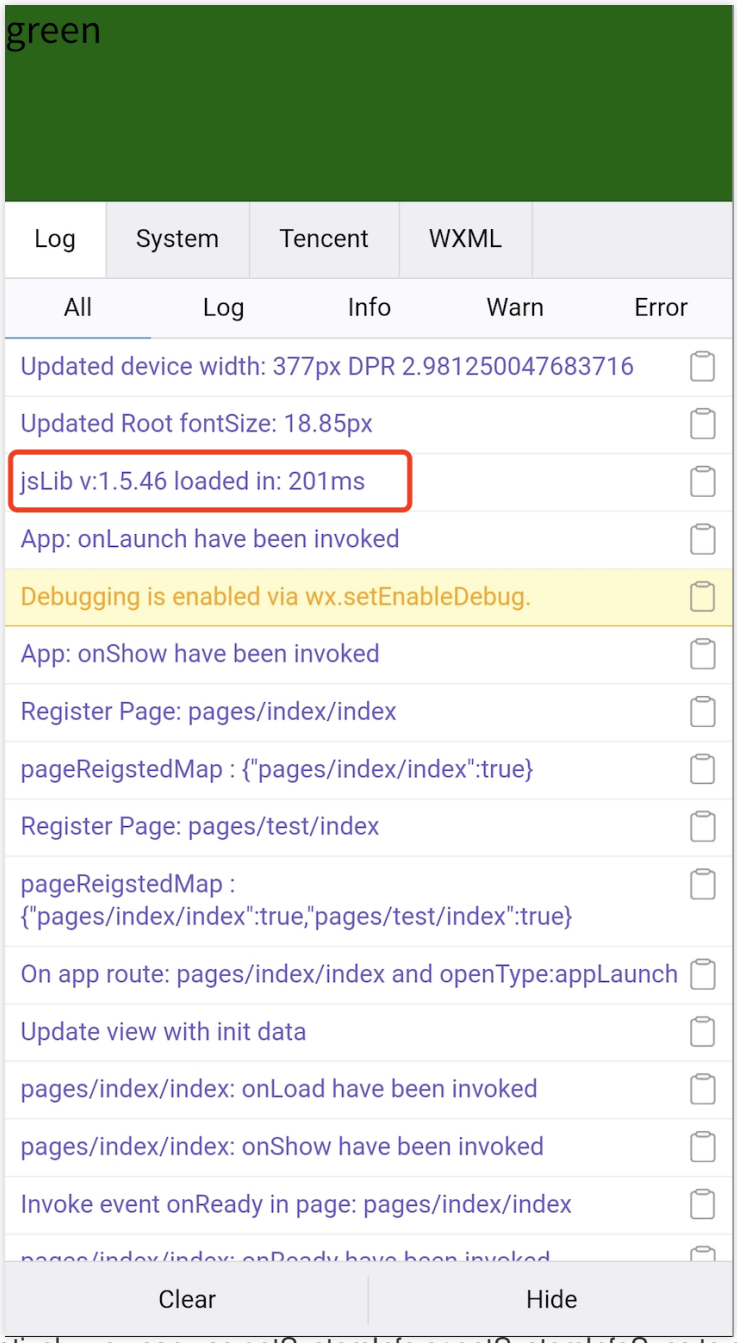
App：After the base library update is completed, the SDK will check for a new version of the base library the next time you open a mini program. If a new version is detected and the app is running in the foreground, the new version of the base library will be automatically downloaded. The next time you open the mini program, the new version of the base library will be used.

IDE：You can switch to the corresponding version of the base library on the project development panel.

3. **How to check the version of the base library in use?**

If the mini program is in debug mode, you can see the base library version information at the top of the logs.

©2013-2025 Tencent Cloud International Pte. Ltd.



Alternatively, you can use getSystemInfo or getSystemInfoSync to check the base library version information.

IDE Operation Instructions

Overview

Last updated：2025-01-16 19:36:24

Our IDE is a cross-platform desktop application, designed to assist developers in accomplishing the development, preview, and debugging of Mini Programs. Upon completion of debugging and previewing, developers are required to conduct regression testing on real devices, as certain device APIs (such as Bluetooth, NFC, WiFi, etc.) need such regression testing.

Supported Platforms and Acquisition

Our Mini Program Developer Tools support common mainstream platforms. For the download process, see Getting Started.

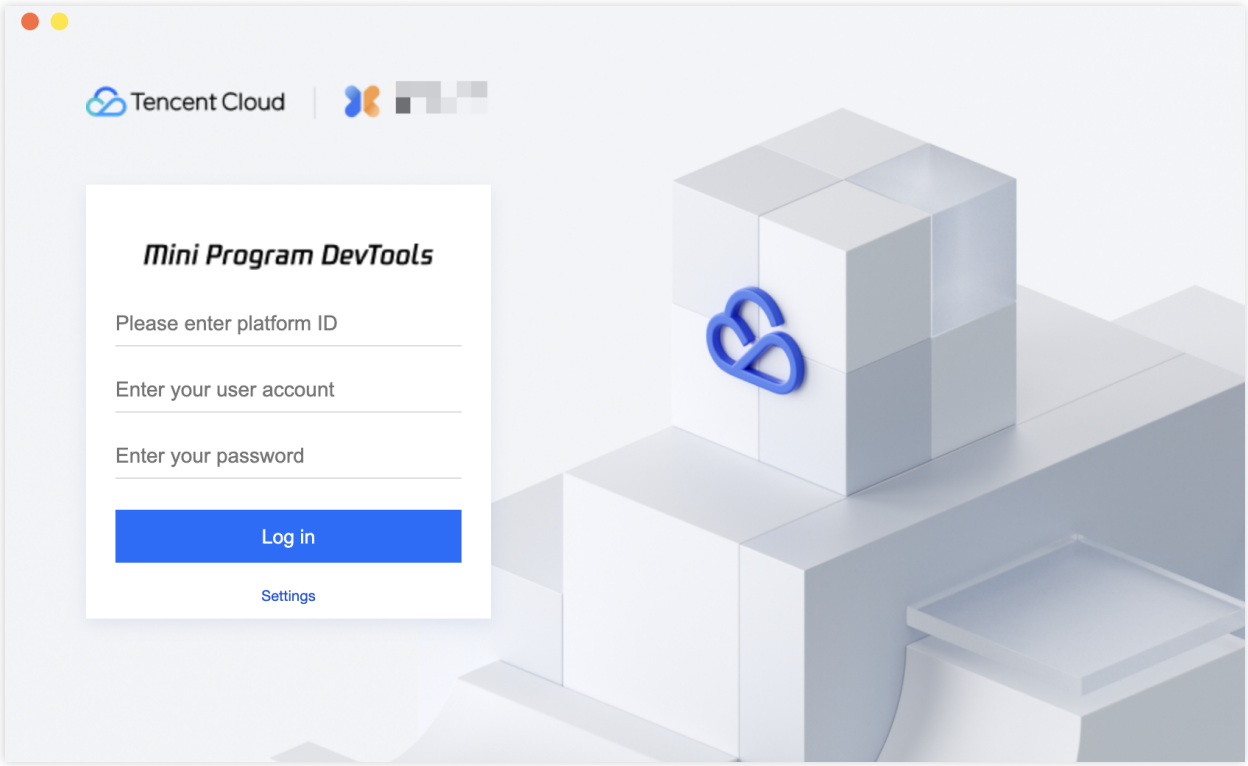
©2013-2025 Tencent Cloud International Pte. Ltd.

Interfaces

Last updated：2025-01-23 17:22:59

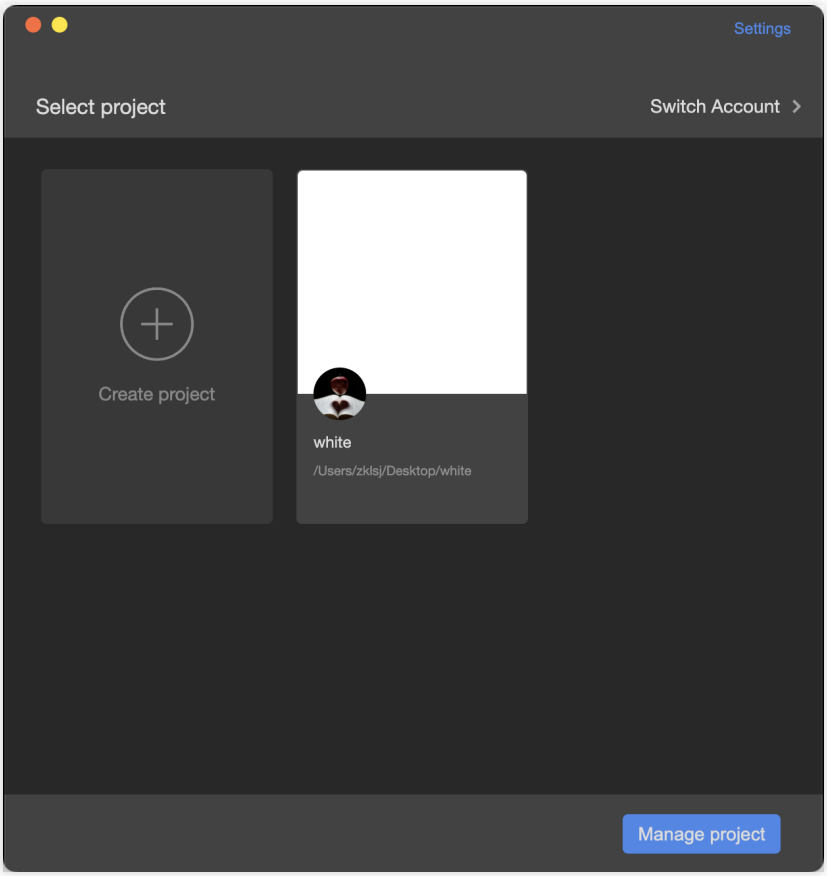
Startup page

On the login page, users need to enter the platform ID (to distinguish different tenants), account, and password to log in. For branded versions of the IDE, users can log in directly with their account and password.



Project list

After a successful login, you will see a list of existing projects.



Creating a project

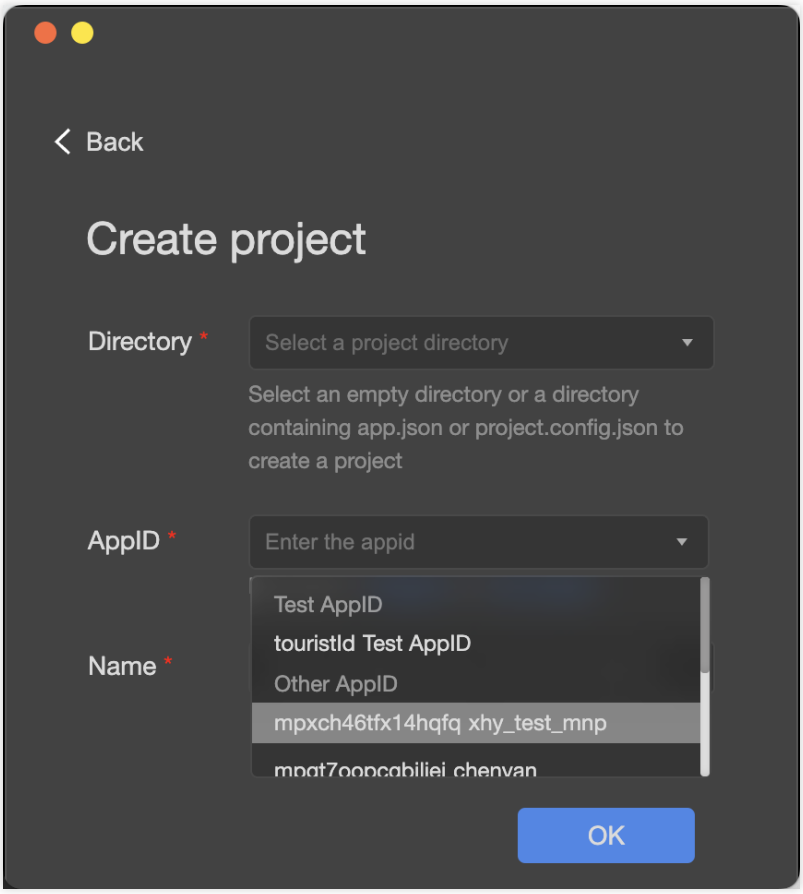
To create a local project, select an empty folder. A simple default project will be generated in this directory. If the folder contains app.json, the project will be opened directly.

When creating a new project, after confirming the path, you also need to fill in the AppId and local name.

The AppId can be manually entered or selected from the current user's existing mini program permissions, or you can use a test mini program AppId.

The name can be anything you choose (for local distinction).

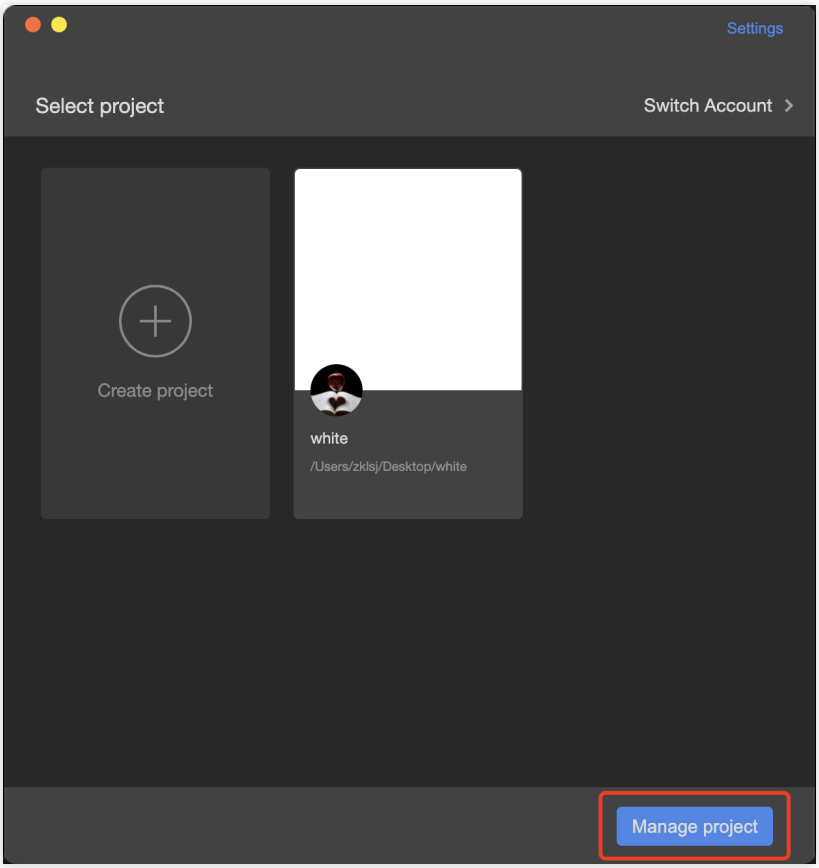
©2013-2025 Tencent Cloud International Pte. Ltd.



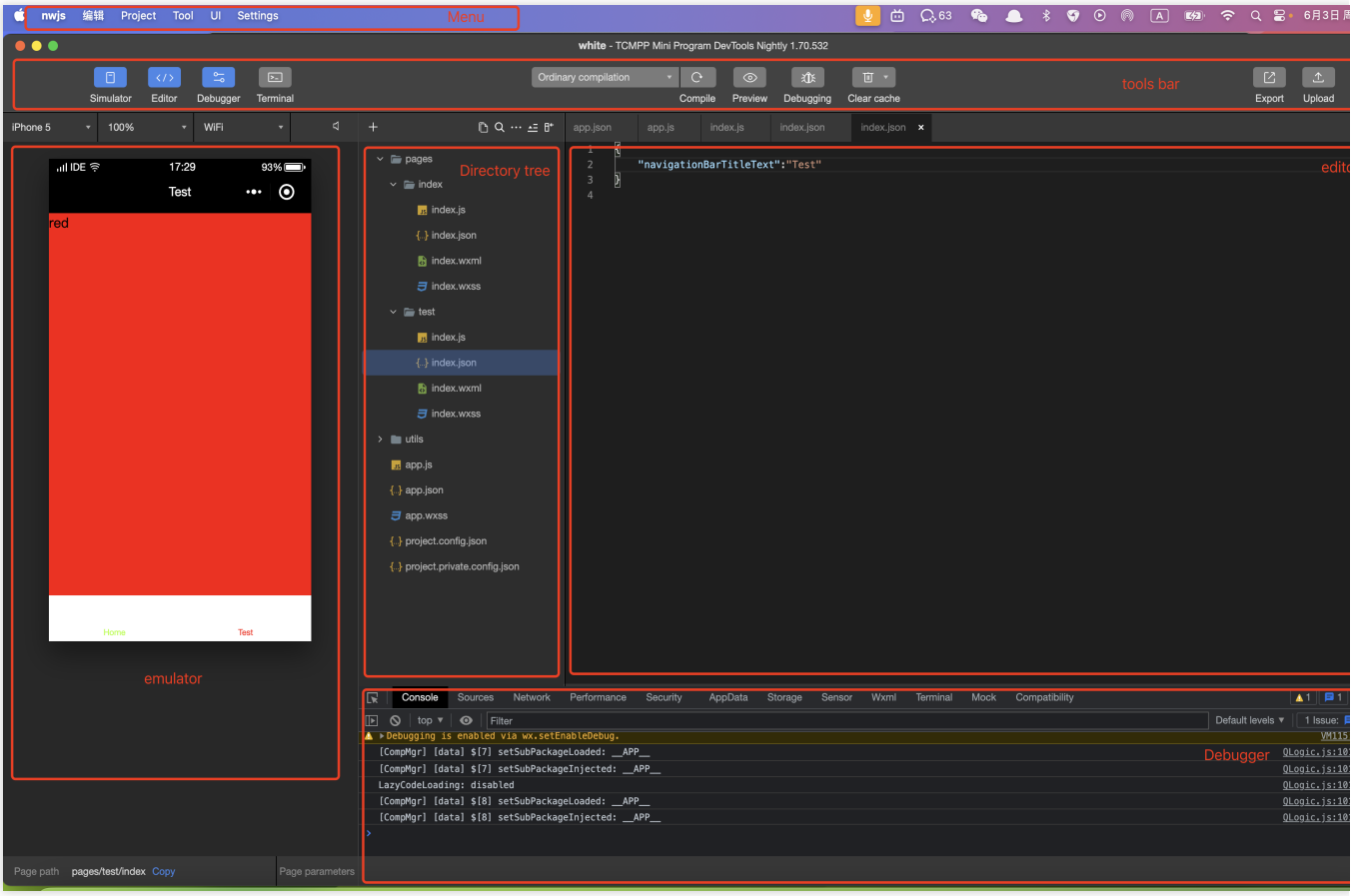
Project management

You can delete or batch delete local projects (this will not affect the actual disk files, only the local project configuration).

Home page


Settings
Select project
Switch Account
Create project
white
/Users/zklsj/Desktop/white
Manage project

Home page



Menu bar

Devtools

Switch account: Quickly switch the logged-in user.

About: Information about the developer tools.

Check for updates: Check for version updates.

Debug: Debug the developer tools and editor. If you encounter a suspected bug in the developer tools or editor, you can open the debug tools to check for error logs and provide feedback.

Restart: Restart the IDE.

Exit: Close the developer tools.

Editing

System built-in menu for common text editing operations.

Projects

Create a new project

Open a recent project

View all projects

Close current project

Tools

Compile: Recompile the project.

Refresh: Refresh the project.

Preview: Preview the project and generate a preview QR code.

Upload: Upload the mini program to the backend and open the upload window.

Build npm: Build the current mini program npm package. You can refer to NPM usage for details.

Global search: Entry point for global file search.

UI

Toolbar: Toggle to display and hide.

Simulator: Toggle to display and hide.

Debugger: Toggle to display and hide.

Editor: Toggle to display and hide.

Settings

Language: Switch languages.

Theme: Set theme mode, either dark or light.

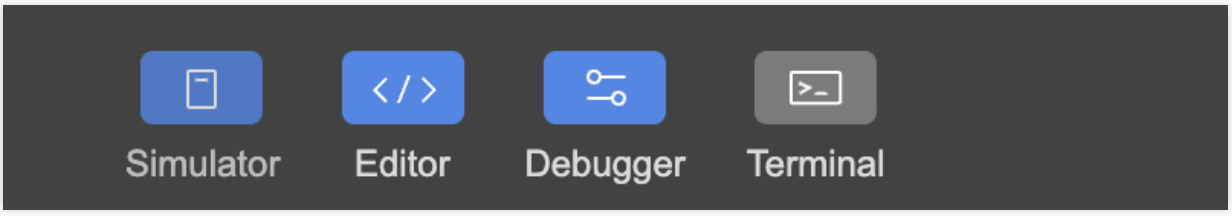
Proxy: Set proxy.

Security settings: Enable server port.

Shortcut settings: Modify shortcuts.

Project settings: View project details panel.

Toolbar



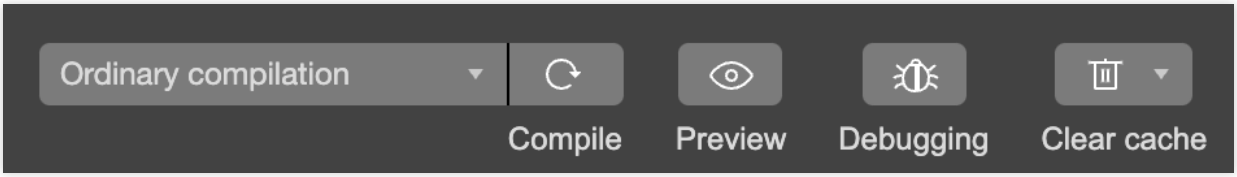
Simulator: Toggle to display and hide the simulator.

Editor: Toggle to display and hide the editor.

Debugger: Toggle to display and hide the debugger.

Terminal: Terminal shortcut.

©2013-2025 Tencent Cloud International Pte. Ltd.

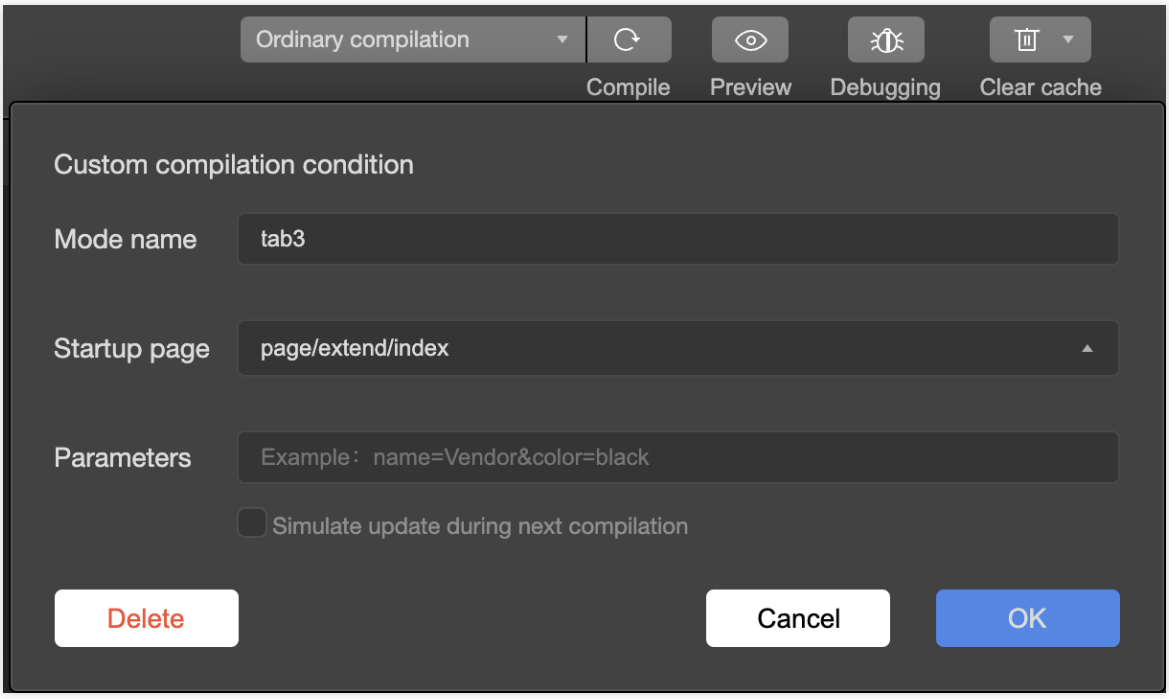


Compile: Recompile the mini program.

Preview: Generate a preview QR code.

Real Device Debugging: Generate a real device debugging code, then scan with a real device to debug.

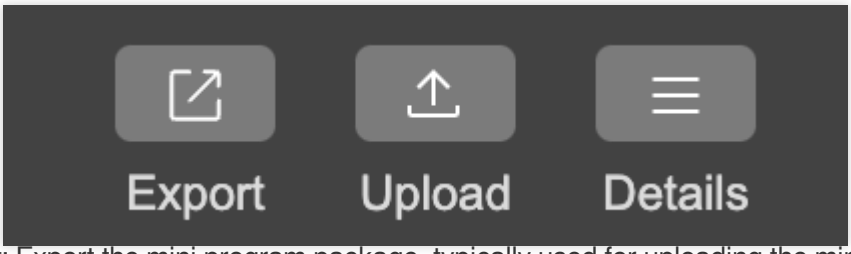
Clear Cache: Conveniently clear file cache, data cache, and backend authorization data on the tool, making it easier for developers to debug.



Add compilation mode: Quickly open a specified page and configure its startup parameters. Once this configuration is set, it will be saved in the project's project.config.json file. It is recommended to commit this file to your Git repository for easy switching of compile modes in the future.

Simulate update during next compilation: Automatically triggers an update request once, which will be automatically canceled the next time.

©2013-2025 Tencent Cloud International Pte. Ltd.

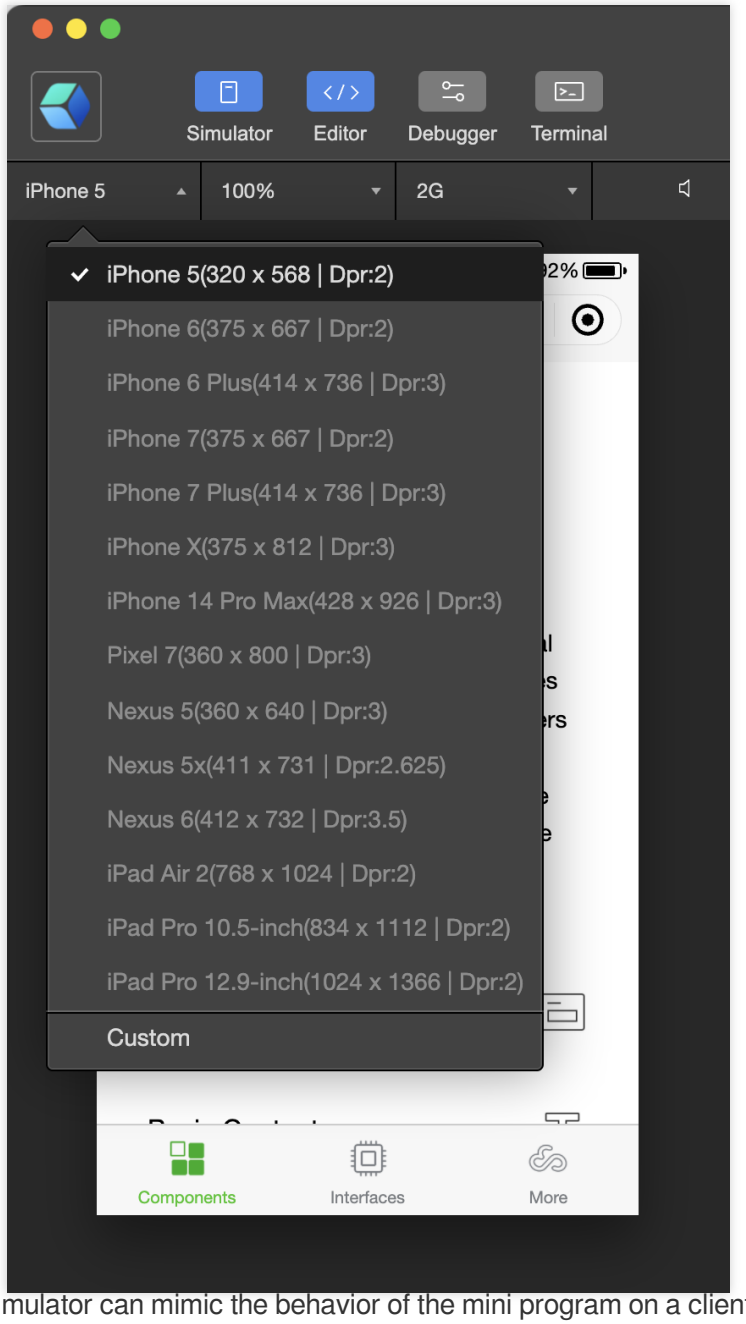


Export: Export the mini program package, typically used for uploading the mini program to the console.

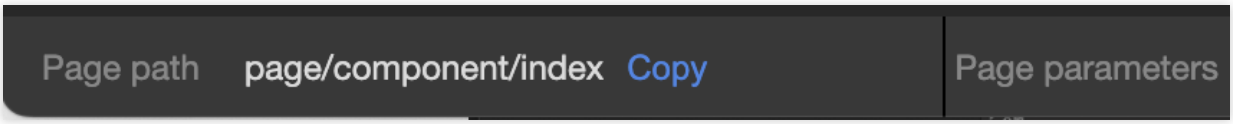
Upload: Directly upload the current mini program to the backend.

Details: View the mini program details panel.

Simulator



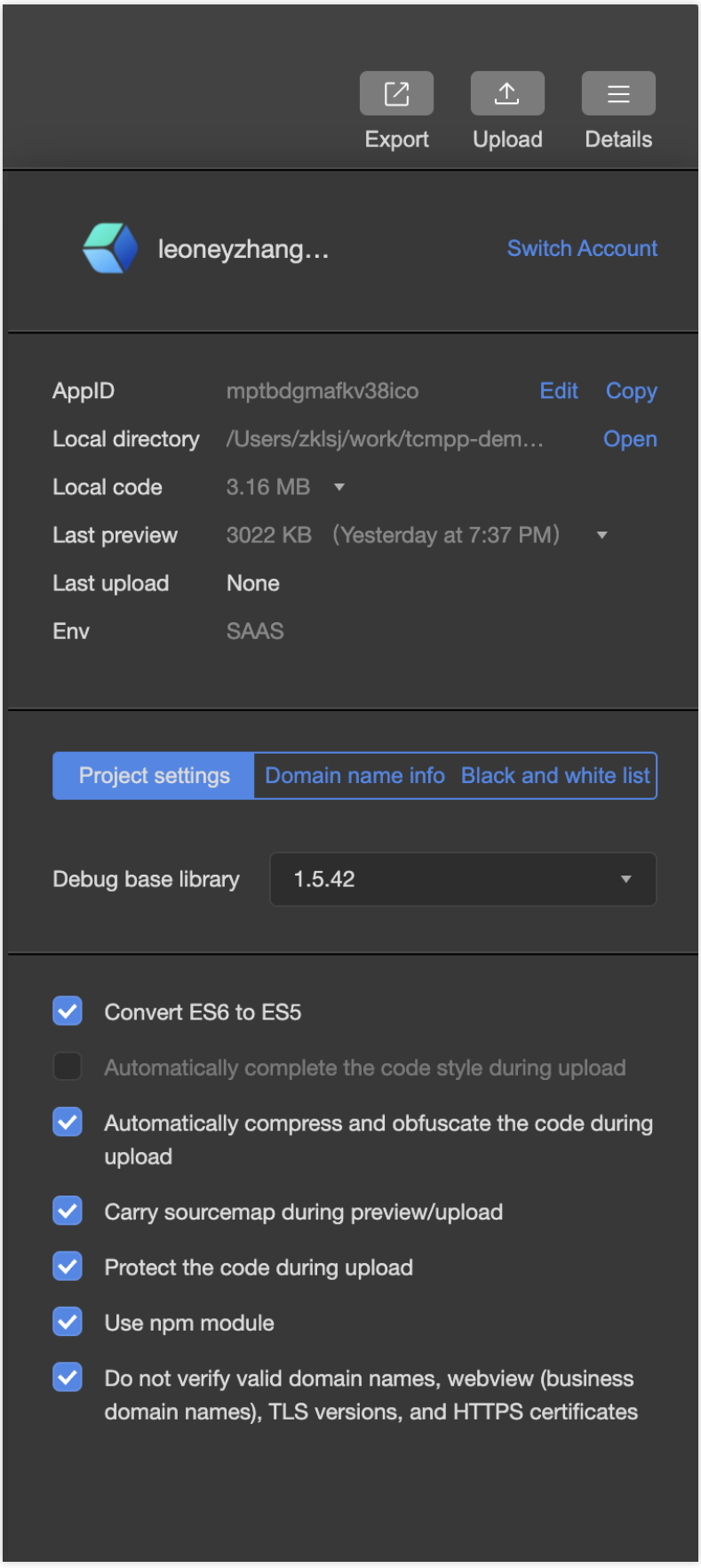
The simulator can mimic the behavior of the mini program on a client. After compiling the mini program's code, it can run directly in the simulator. Developers can choose different devices or add custom devices to debug how the mini program adapts to different screen sizes.



At the bottom status bar of the simulator, you can clearly see the current page path and page parameters of the running mini program.

Project tab

©2013-2025 Tencent Cloud International Pte. Ltd.


Export
Upload
Details
leoneyzhang…
Switch Account
AppID
mptbdgmafkv38ico
Edit
Copy
Local directory
/Users/zklsj/work/tcmpp-dem…
Open
Local code
3.16 MB
Last preview
3022 KB (Yesterday at 7:37 PM)
Last upload
None
Env
SAAS
Project settings
Domain name info
Black and white list
Debug base library
1.5.42
Convert ES6 to ES5
Automatically complete the code style during upload
Automatically compress and obfuscate the code during upload
Carry sourcemap during preview/upload
Protect the code during upload
Use npm module
Do not verify valid domain names, webview (business domain names), TLS versions, and HTTPS certificates

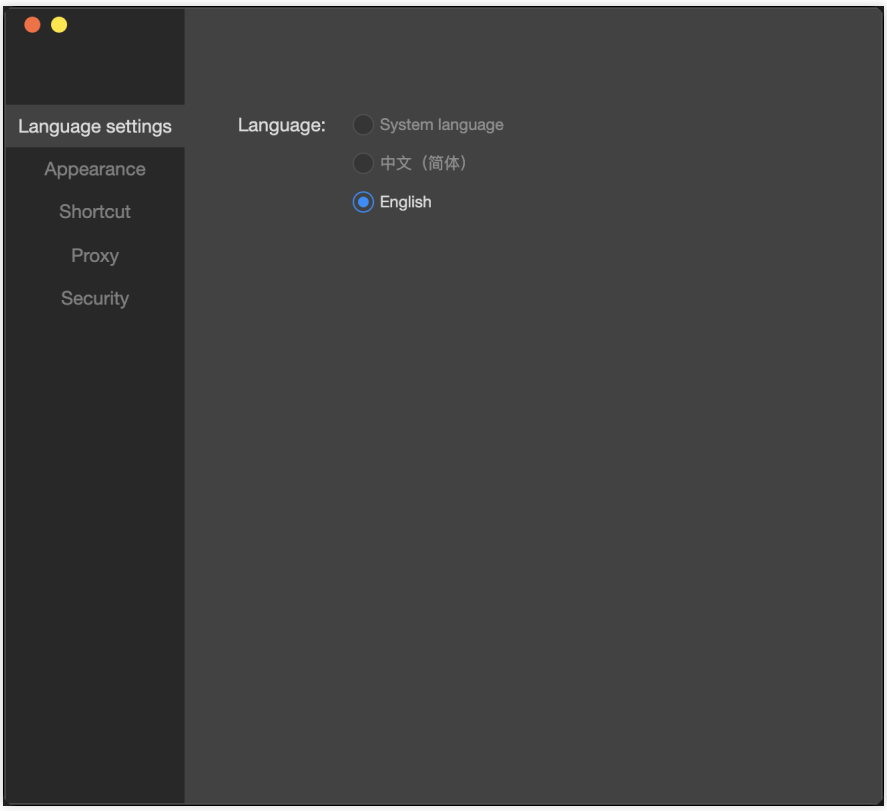
The current panel displays basic information about the mini program, such as AppId, local directory, local code size, last preview, upload information, and current environment information.
Additionally, it shows the base library version, allowing developers to switch between different versions for development and adaptation.
Under domain information, developers can refresh the backend-configured domains. This feature works in conjunction with "Do not verify valid domains, WebView (business domains)." Normally, the request domain and WebView business domain of the mini program need to be configured in advance before going live. For details, see the domain information configuration guide.
The blocklist/allowlist for the same domain information is controlled by the platform. For details see the configuration and usage documentation.

©2013-2025 Tencent Cloud International Pte. Ltd.

Settings

Last updated：2024-11-21 18:49:40

The IDE settings page allows you to manage and configure various global settings, including theme, language, shortcuts, and security proxies.



Language settings

Set the language for the current editor, supporting Chinese and English, with the default following the system language.

Editor theme

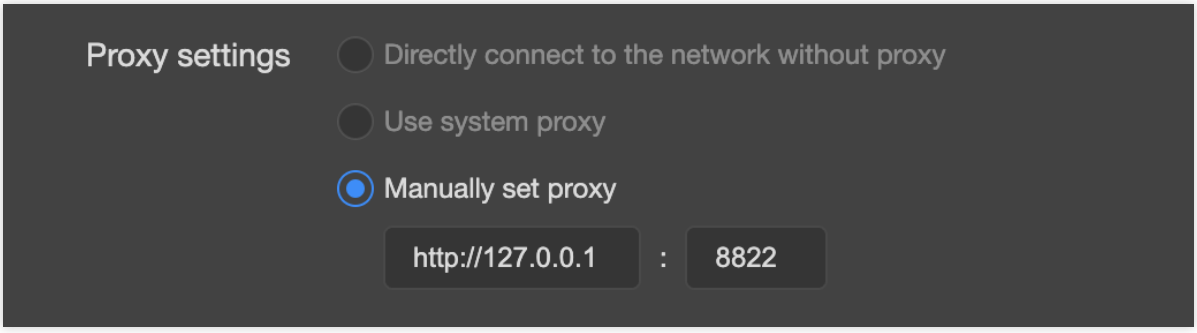
Set the IDE theme, supporting both dark and light mode.

Shortcuts

Include interface toggles and tool shortcuts
for showing and hiding the simulator, debugger, editor, and toolbar.

Proxy

Users can choose to use the system proxy, no proxy (direct connection), or manually set a proxy.
In special network environments (e.g., intranet), you may need to configure a special proxy to access the public network. Manual proxy settings can be configured for such cases.



Code Editing

Last updated：2024-11-28 10:47:59

The editing area allows you to write code, add, delete, and rename files for the current project.

File format

Since iOS only supports UTF-8 encoding, the latest version of the developer tool will check the encoding format of code files during upload.

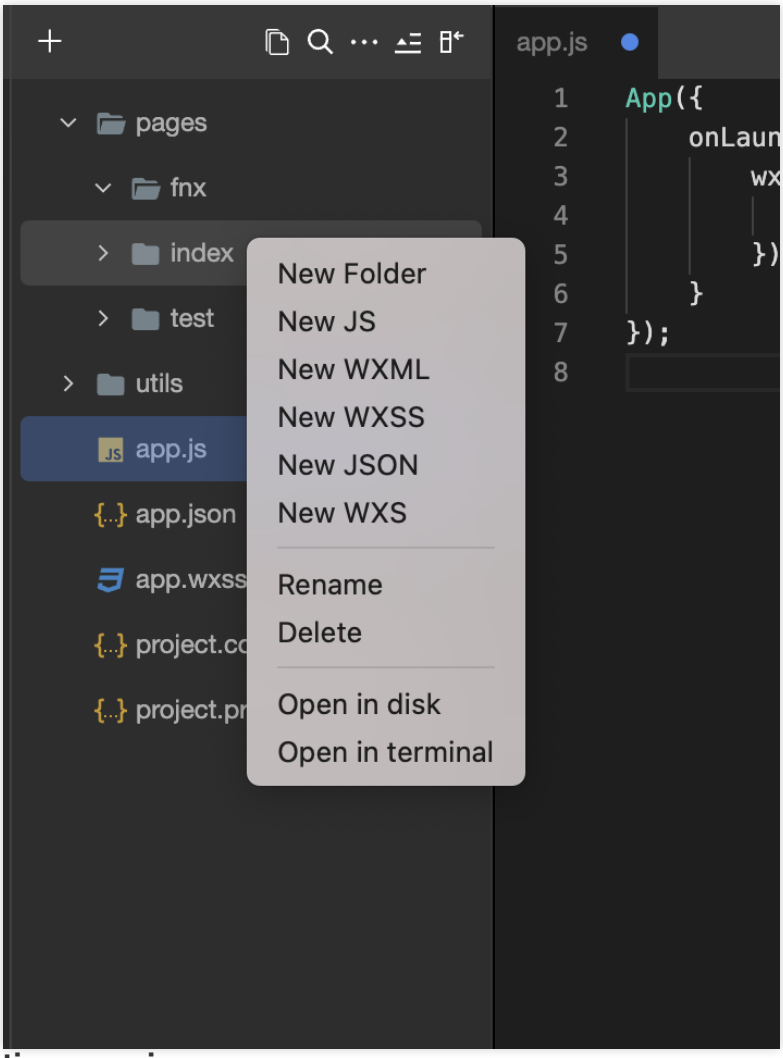
Supported files

The tool currently supports editing five types of files: WXML, WXSS, JS, JSON, WXS, and image file previews.

File operations

Users can right-click on the file tree to quickly create new folders and files.

©2013-2025 Tencent Cloud International Pte. Ltd.



Real-time preview

When WXML, WXSS, JS, OR JSON files are modified, the mini program will automatically trigger a compilation, and the changes will be previewed in real-time in the simulator.

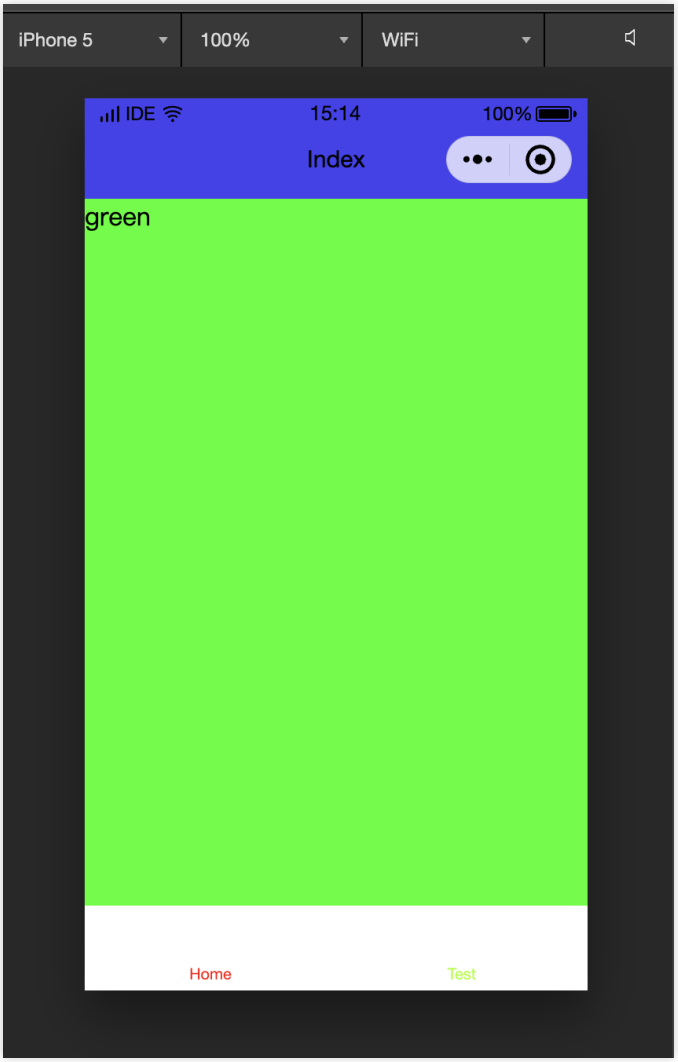
Mini Program Debugging

Last updated：2024-11-28 10:48:23

The main debugging areas include the simulator, debugging tools, and the mini program operation area.

Simulator

The simulator mimics the real behavior of the mini program on a client, accurately presenting most APIs.



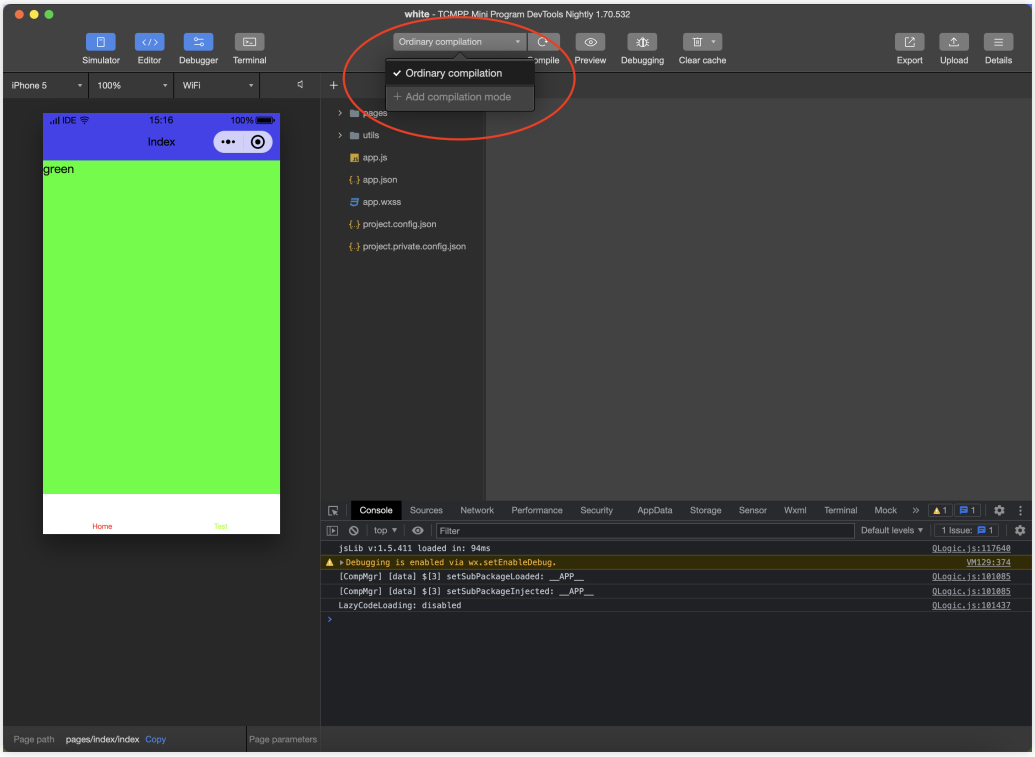
Custom compilation

Click the compile button in the toolbar or use the shortcut Ctrl (⌘) + B to compile the current code and automatically refresh the simulator.

Developers can add or select existing custom compile conditions to debug specific pages with different parameters.

Note：

Compile conditions are project-specific and can be saved for each project.



Debugging tool

The debugging tools are divided into nine functional modules: WXML, Console, Sources, Network, Appdata, Storage, Sensor, Mock, and Compatibility.

Wxml panel

This helps developers see the transformed Wxml interface, showing the real page structure and corresponding WXSS properties. You can modify WXSS properties and see changes in real-time in the simulator (changes are not saved to the file). The selector in the top left of the debugging module allows quick location of WXML code for page components.

Console panel

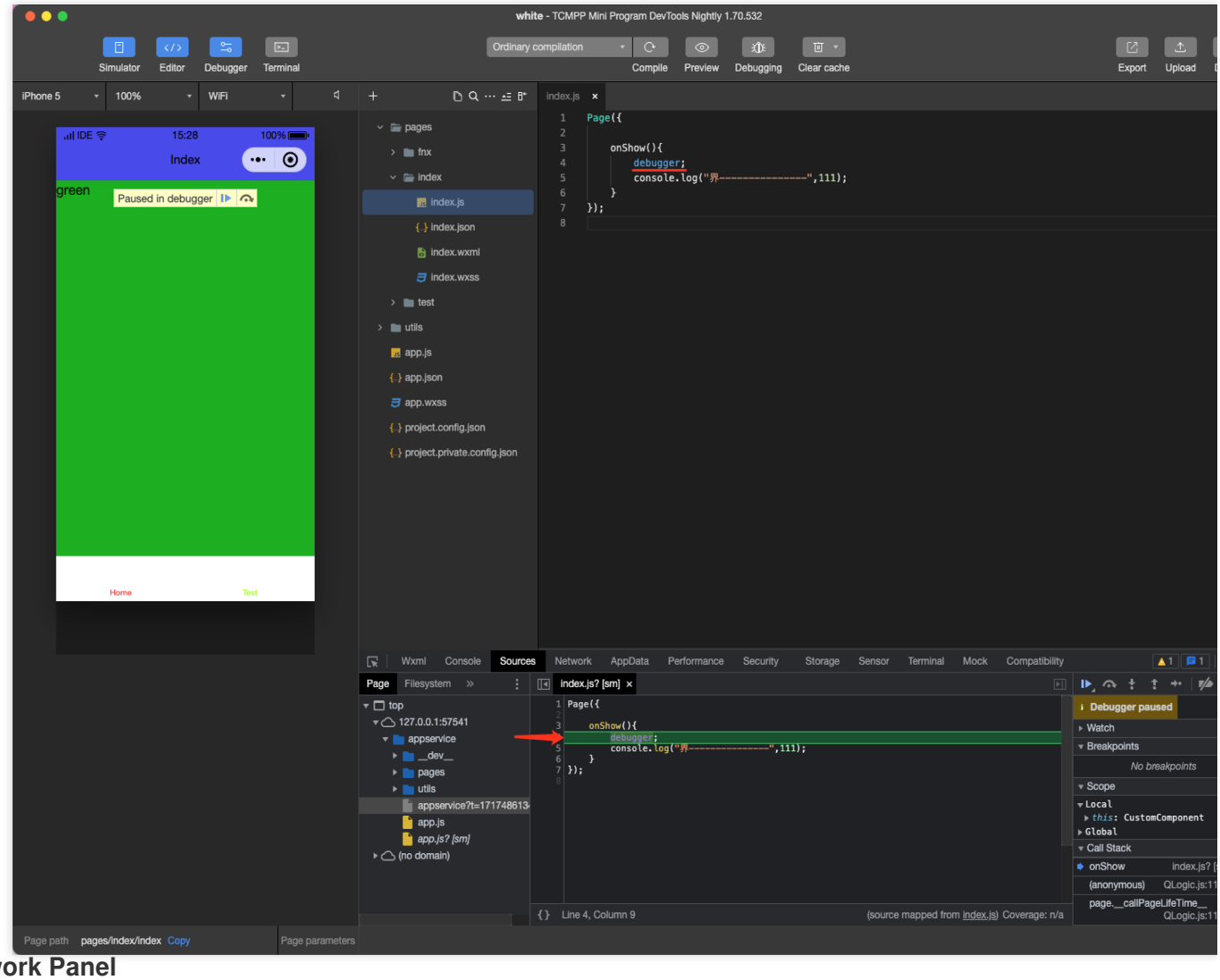
The Console panel has two main features:

Developers can input and debug code here.

Errors from the mini program are displayed here.

Source panel

This is used to display the script files of the current project. Unlike browser development, the mini program framework compiles script files, so the files shown in the Sources panel are processed scripts. Developer code is wrapped in a define function, and Page code includes a require call at the end.
Developers can add debugger in the code to set breakpoints.



Network Panel

The Network Panel is used to observe and display request and socket request statuses.

Appdata Panel

The AppData panel is used to show the specific data of the mini program's AppData during runtime, reflecting project data in real-time. You can edit data here, and it will be immediately reflected on the interface.

Storage Panel

Storage panel is used to display data stored using wx.setStorage or wx.setStorageSync.
You can delete (using the delete key), add, or modify data directly in the Storage panel.

©2013-2025 Tencent Cloud International Pte. Ltd.

Sensor Panel

The Sensor panel has two main features:

Developers can simulate geographic location.

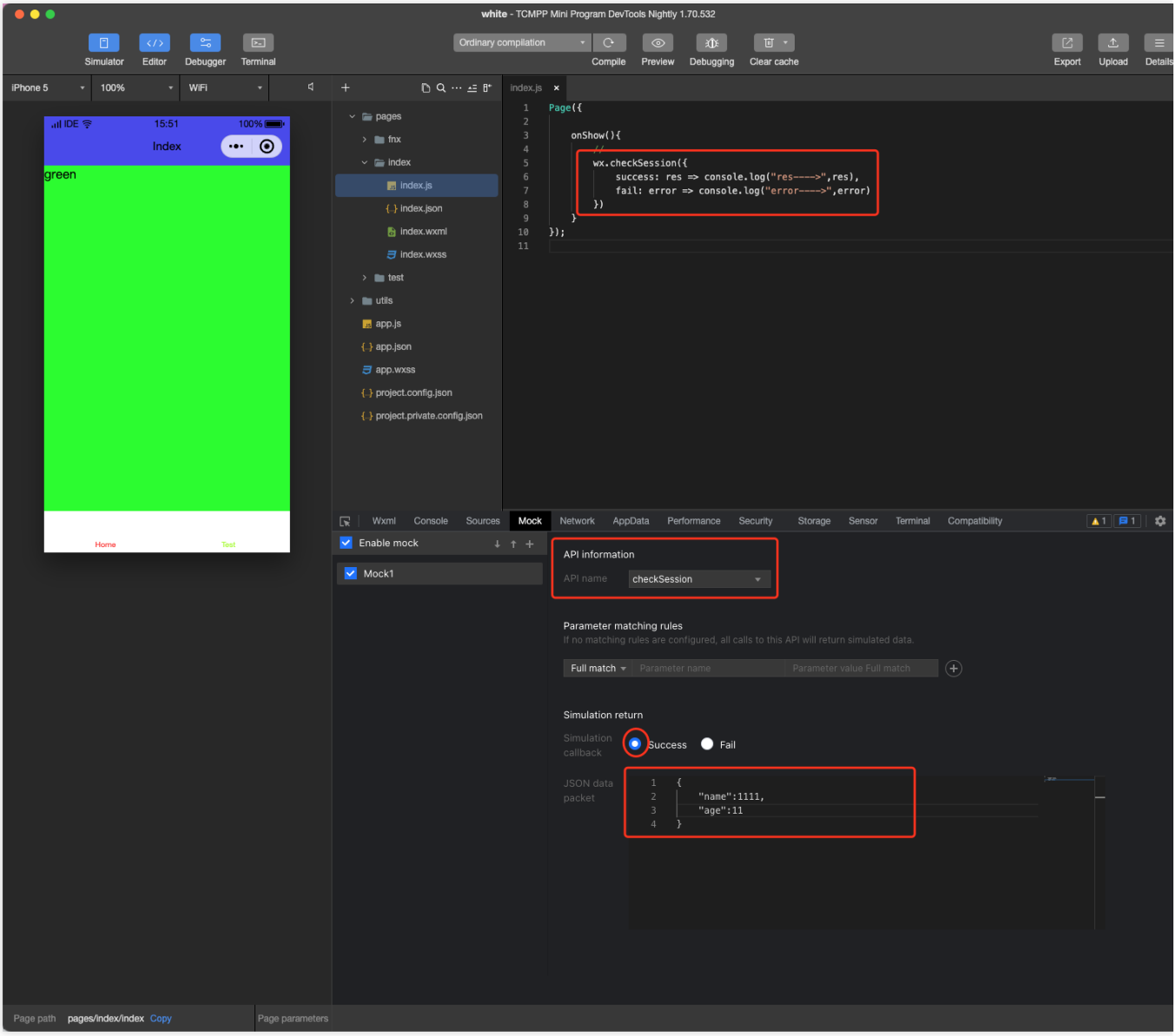
Developers can simulate mobile device behavior for debugging gravity sensor APIs.

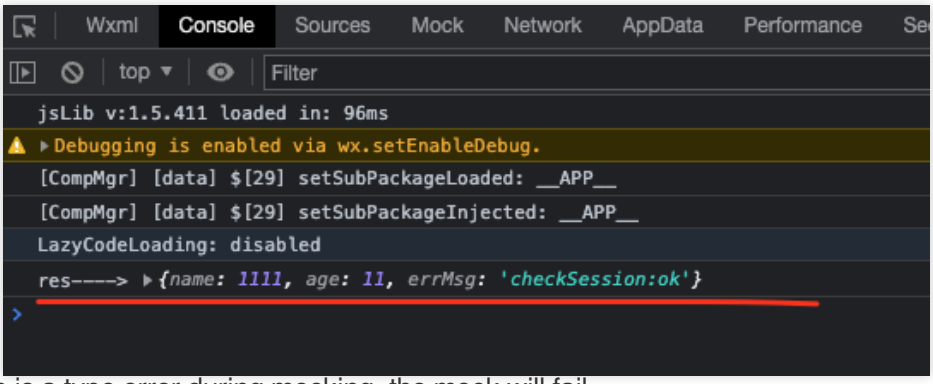
Mock Panel

The developer tool provides API Mock capabilities to simulate API call results, useful for developing mini programs.

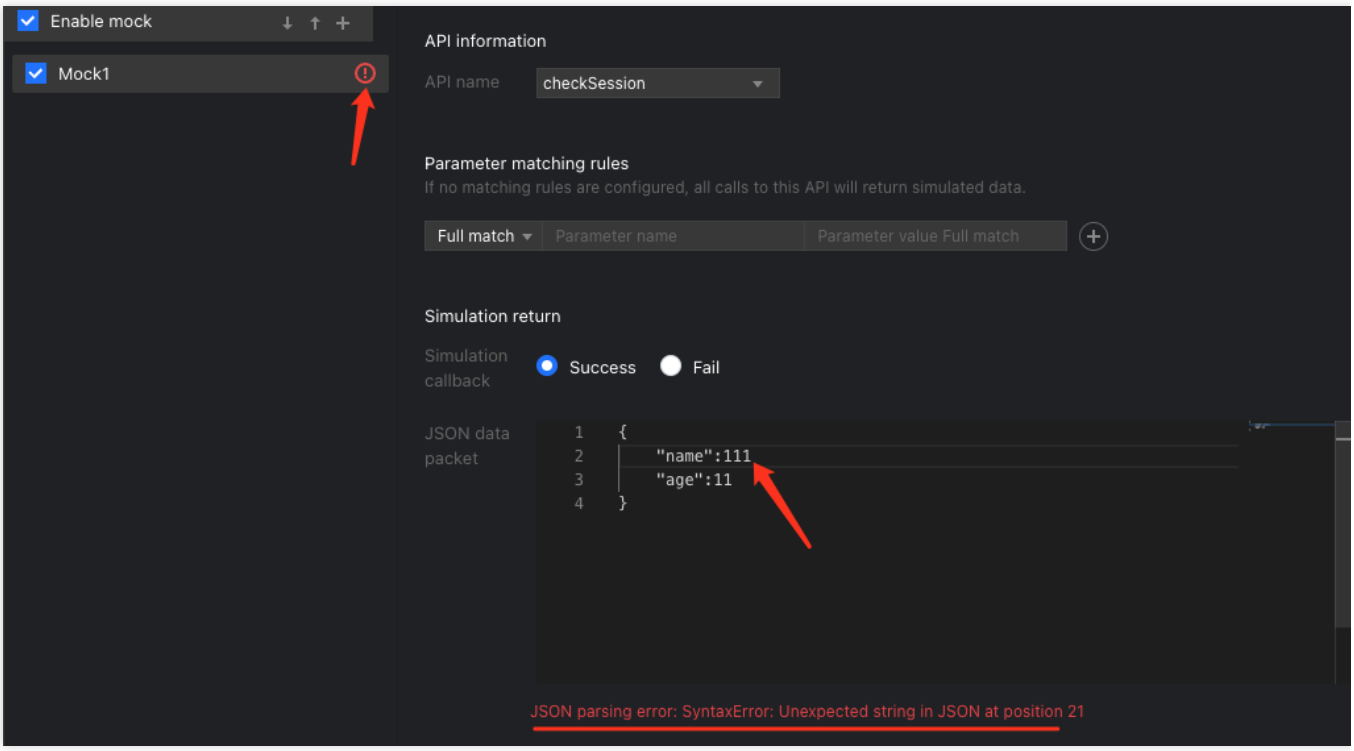
Developers can mock custom APIs or user information.

In the following snippet, the checkSession API will be mocked.





If there is a type error during mocking, the mock will fail.



Compatiblity Panel

Mini program migration tool - Compatibility

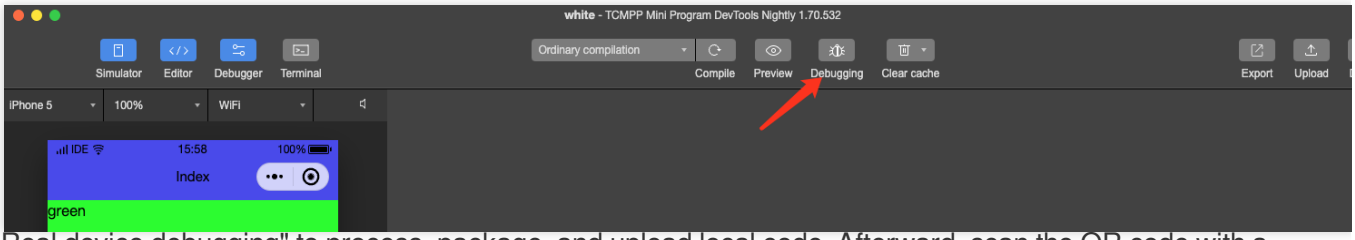
You can initiate Weixin mini program compatibility checks.

The tool will traverse components, APIs, and JSON files in the mini program, displaying results in the lower window.

You can export the results as an Excel file.

Remote debugging

Real device debugging:



Click "Real device debugging" to process, package, and upload local code. Afterward, scan the QR code with a mobile client to open the debugging window and start real device debugging.

Note：

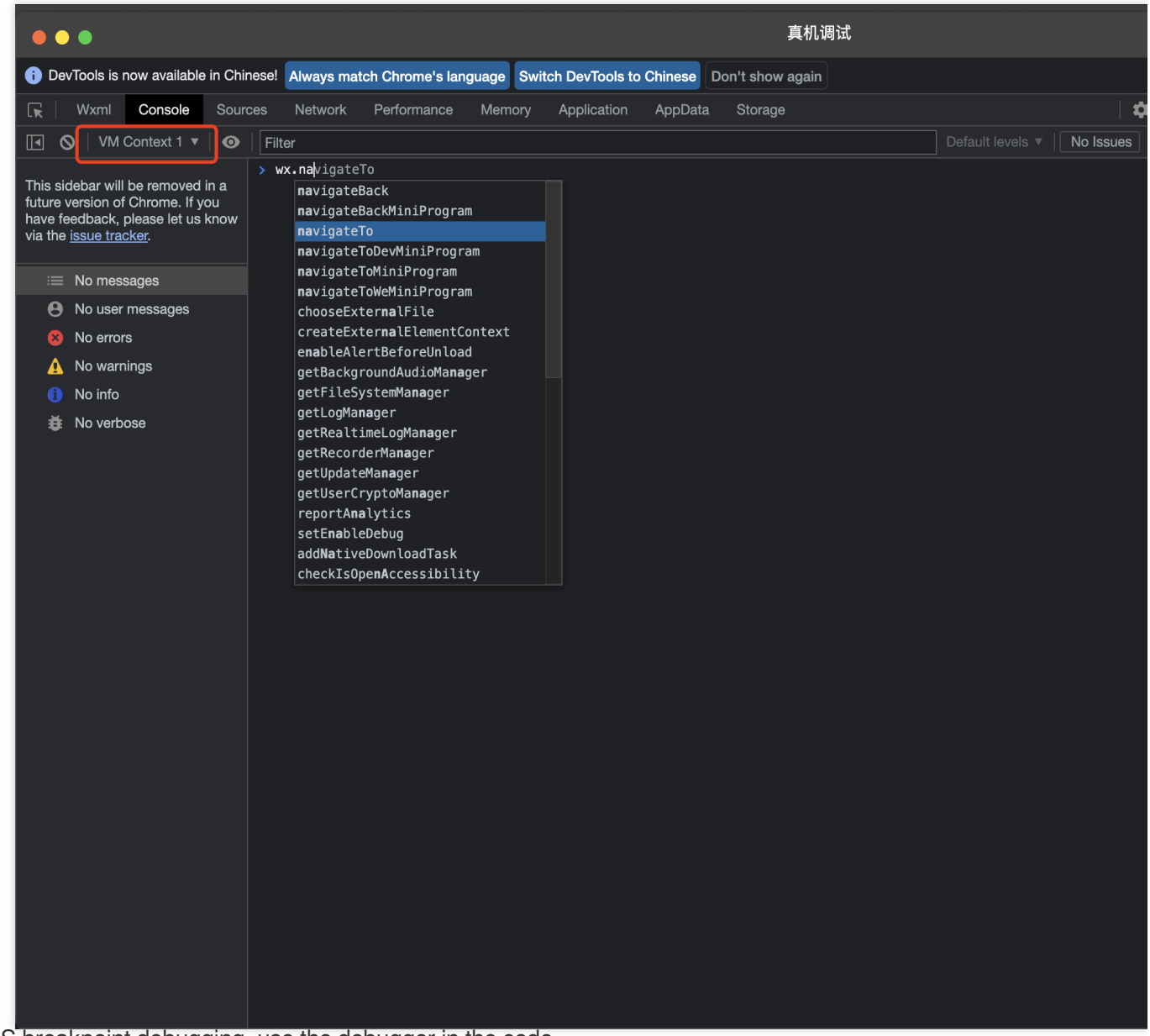
The versions of Android and iOS SDK must be greater than 1.5.12 for real device debugging.

Use an app with the integrated SDK to scan the QR code to start remote debugging.

To end debugging, close the debugging window or click the "End Debugging" button in the lower right corner.

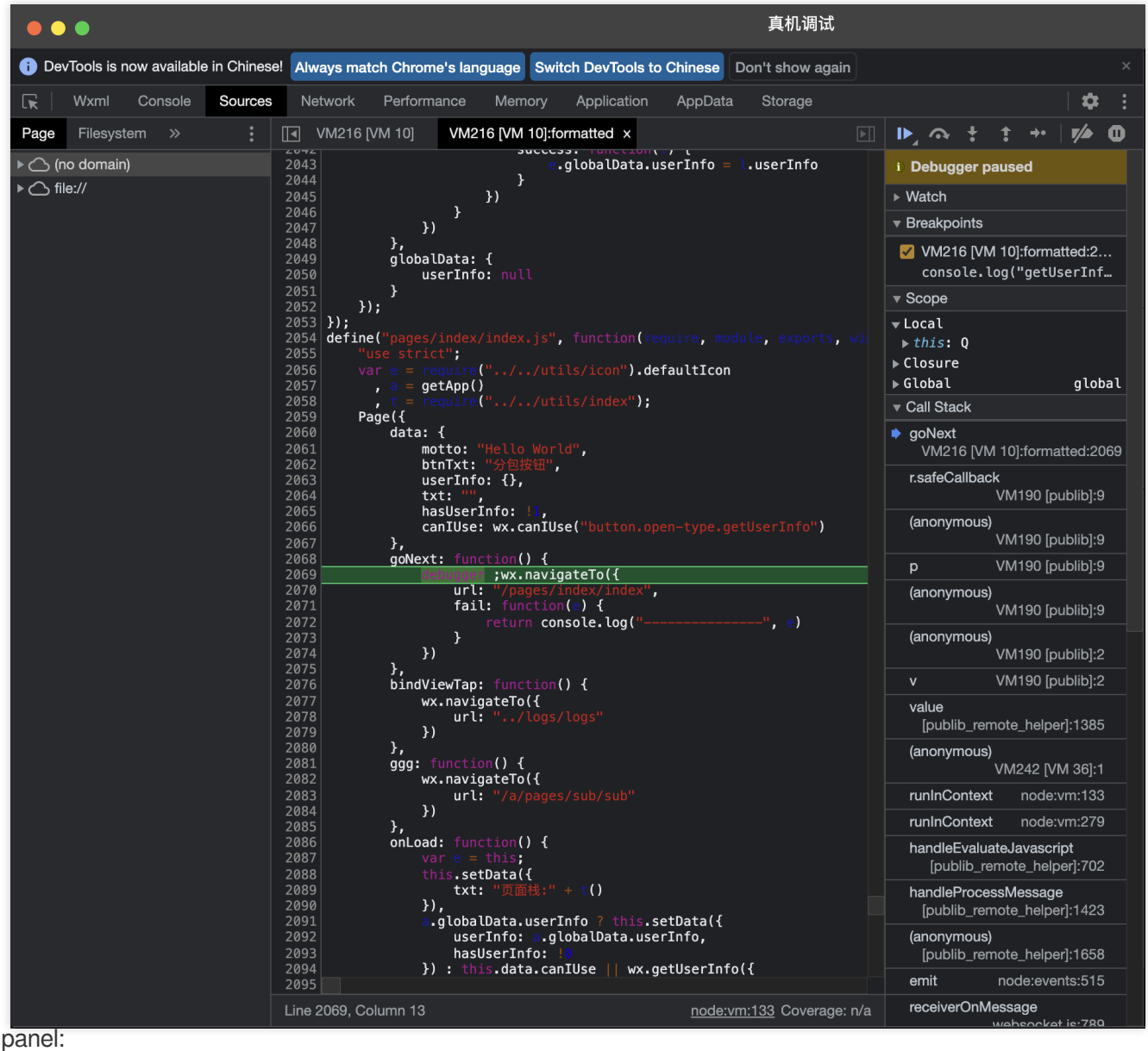
The remote debugging window is divided into two parts: the debugger view on the left and the information view on the right. Developers can debug code directly in the debugger and check Storage status. The information view shows the connection status with the phone and server, and any errors.

In the remote debugger, developers can debug code in the Console panel, view source code and set breakpoints in the Sources panel, and check Storage usage in the Storage panel.

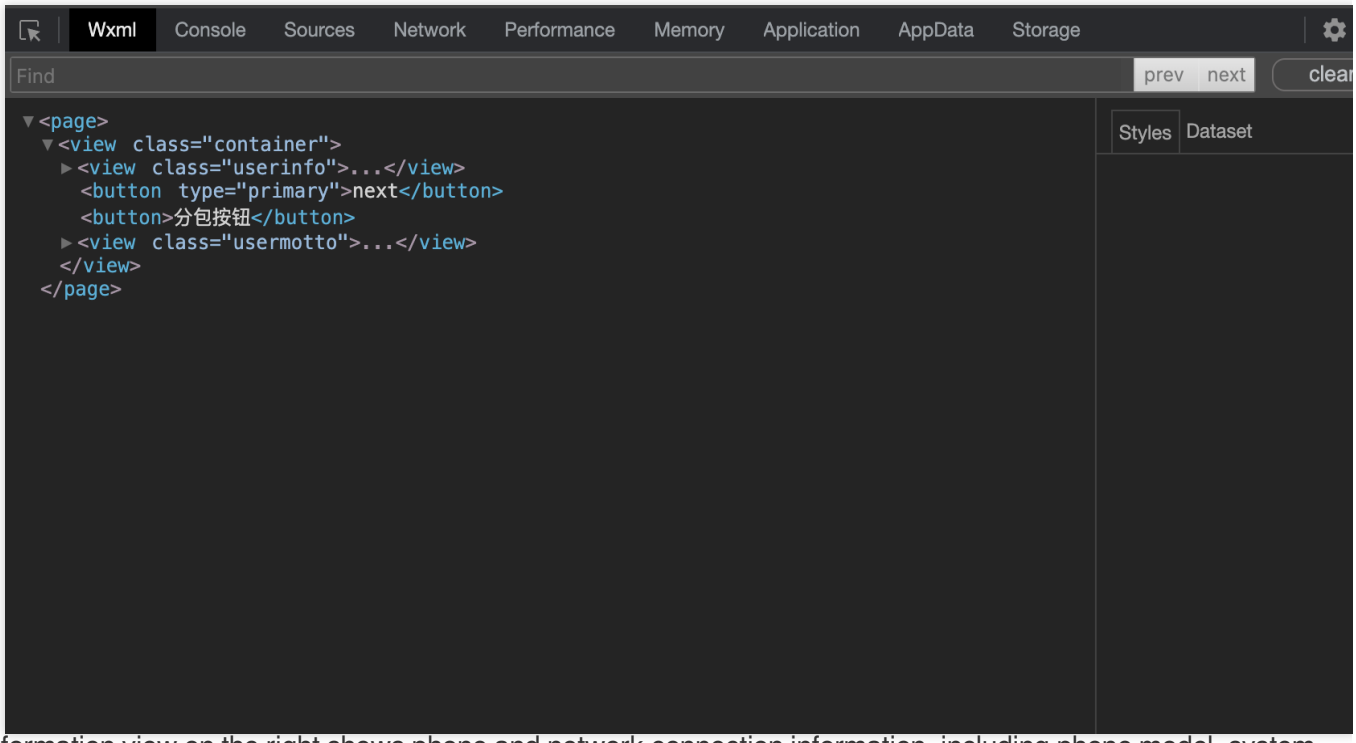


For JS breakpoint debugging, use the debugger in the code.




真机调试
DevTools is now available in Chinese!
Always match Chrome's language
Switch DevTools to Chinese
Don't show again
Wxml
Console
Sources
Network
Performance
Memory
Application
AppData
Storage
Page
Filesystem
(no domain)
file://
VM216 [VM 10]
VM216 [VM 10]:formatted
Debugger paused
Watch
Breakpoints
VM216 [VM 10]:formatted:2...
console.log("getUserInf...
Scope
Local
this: Q
Closure
Global
global
Call Stack
goNext
VM216 [VM 10]:formatted:2069
r.safeCallback
VM190 [publib]:9
(anonymous)
VM190 [publib]:9
p
VM190 [publib]:9
(anonymous)
VM190 [publib]:9
(anonymous)
VM190 [publib]:2
v
VM190 [publib]:2
value
[publib_remote_helper]:1385
(anonymous)
VM242 [VM 36]:1
runInContext
node:vm:133
runInContext
node:vm:279
handleEvaluateJavascript
[publib_remote_helper]:702
handleProcessMessage
[publib_remote_helper]:1423
(anonymous)
[publib_remote_helper]:1658
emit
node:events:515
receiverOnMessage
Line 2069, Column 13
node:vm:133
Coverage: n/a

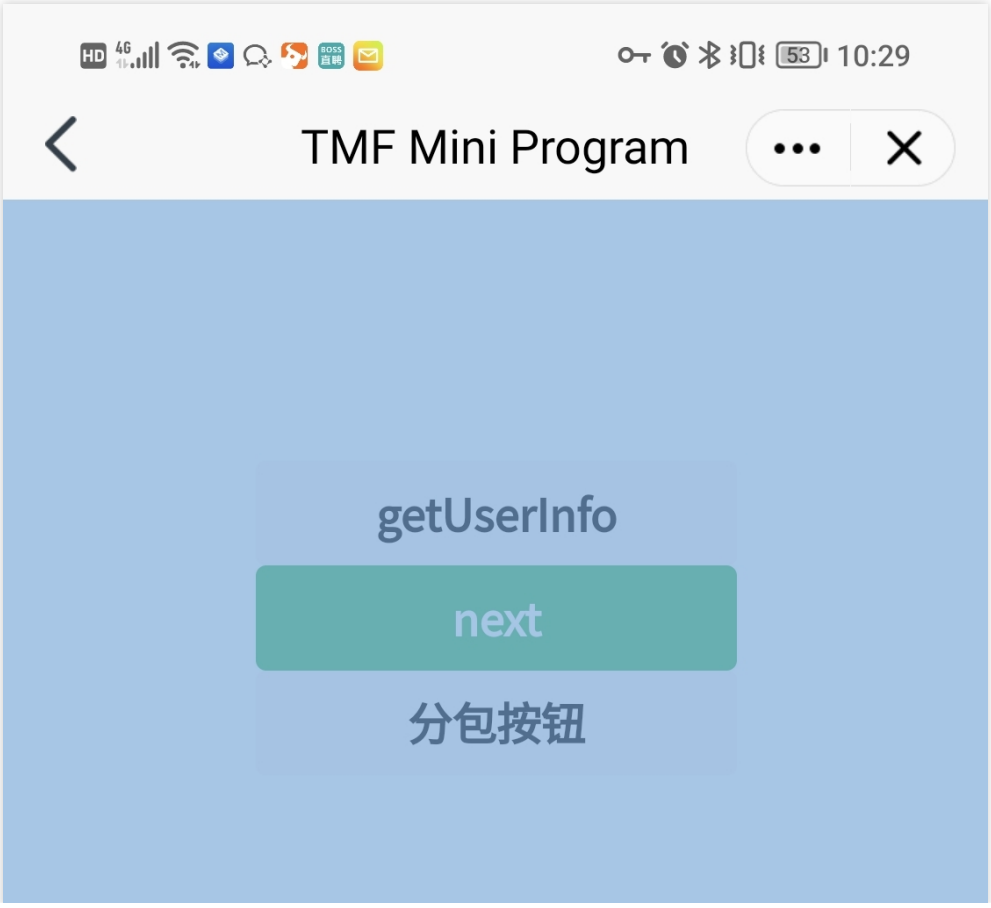
wxml panel:



The information view on the right shows phone and network connection information, including phone model, system, name, host app version, and communication latency. Lower latency means smoother communication with the phone.

Enable node inspection mode to click elements on the real device and view their details.

Display on mobile phones:




已连接
待确认 0
已发送 170
已接收 166
结束
收起
orld
page
377.02 x 547.15
vConsole

Additional Information

Last updated：2024-11-28 10:48:42

Mini program code hardening

Feature overview

To enhance platform security, we address security issues in the mini program development process, such as code being easily decompiled, core business logic being deciphered, and algorithms being repackaged. These issues pose risks of the mini program being cracked and core code being stolen. The platform provides a code hardening feature to encrypt the front-end code of mini programs. This makes it difficult for attackers to read the front-end code logic, while developers should also maintain rigorous development standards to ensure code quality and protect the mini program's security. Code hardening is a feature that allows developers to encrypt the front-end code of mini programs to prevent code exposure.

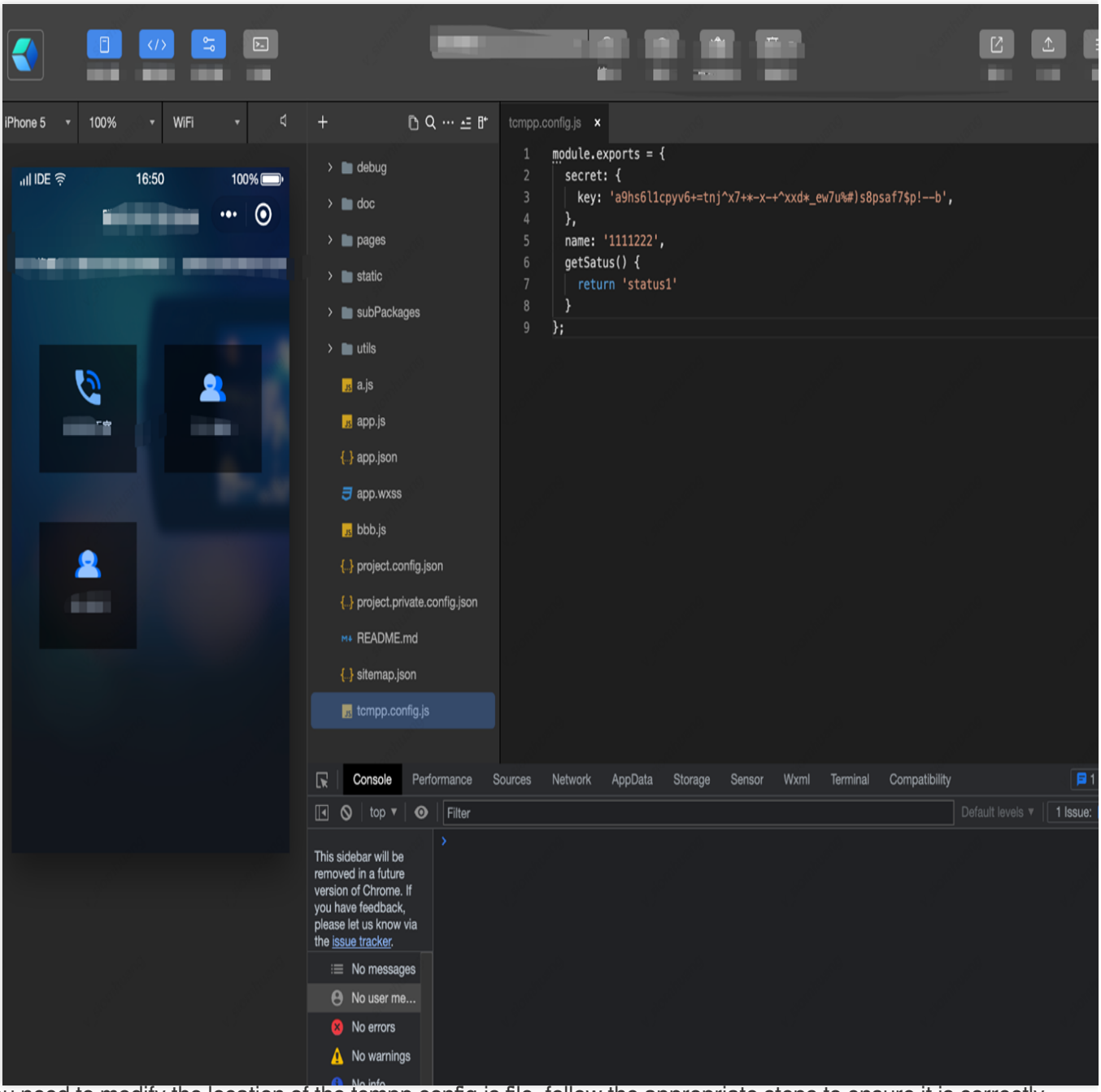
How to use

The code hardening feature allows JavaScript files to be passed to an encryption tool, enabling multiple protection measures such as string encryption, property encryption, call transformation, and code obfuscation. It is recommended to use code obfuscation for sensitive information in the mini program (such as SDK keys), core algorithm logic, version content, and other content that requires security protection.

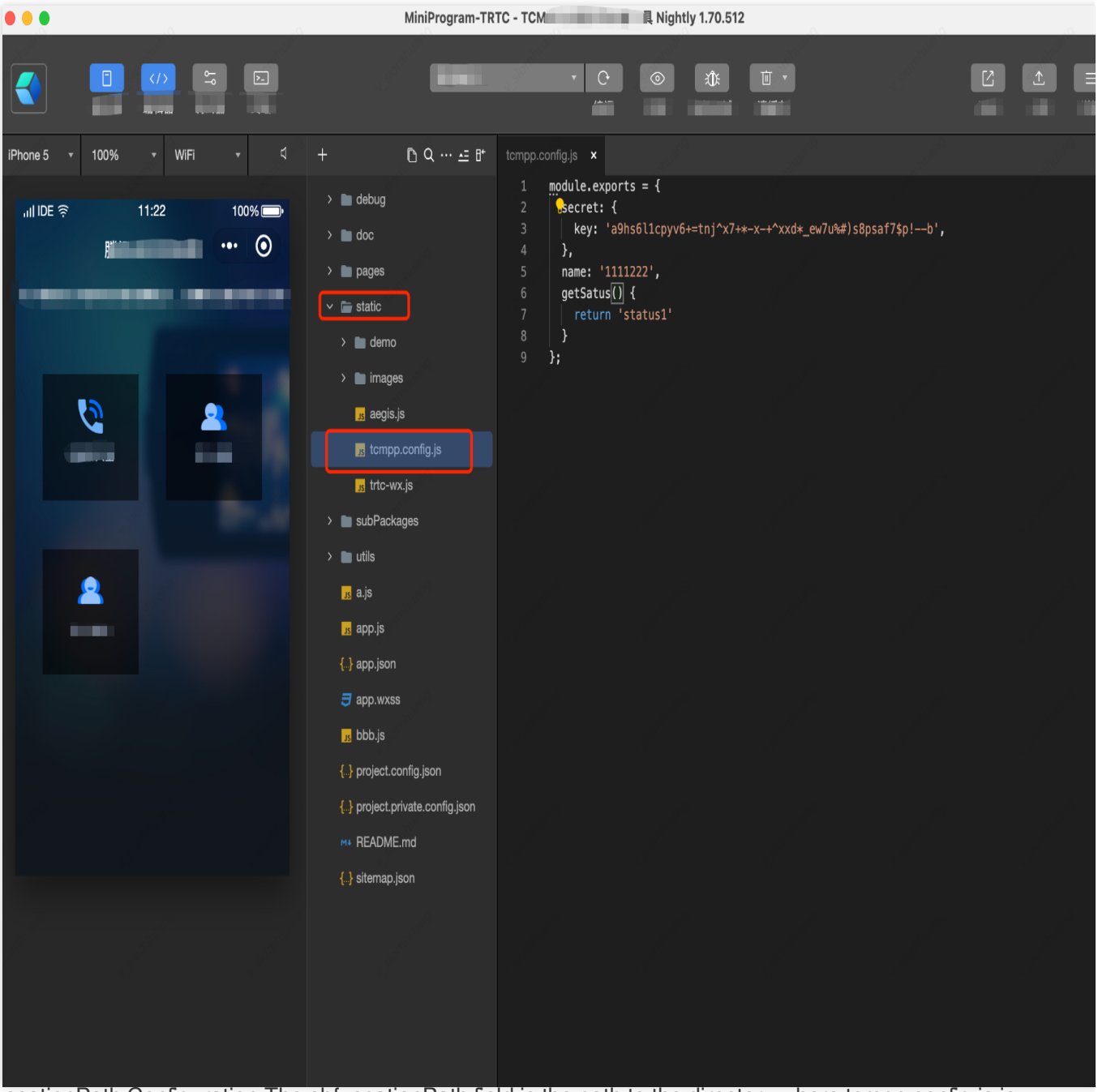
Directions

1. Create a tcmpp.config.js file in the same directory as app.json in the mini program project. You can write sensitive information or core algorithms in this JS file and export them for use. During the compilation phase, tcmpp.config.js will be obfuscated and encrypted.

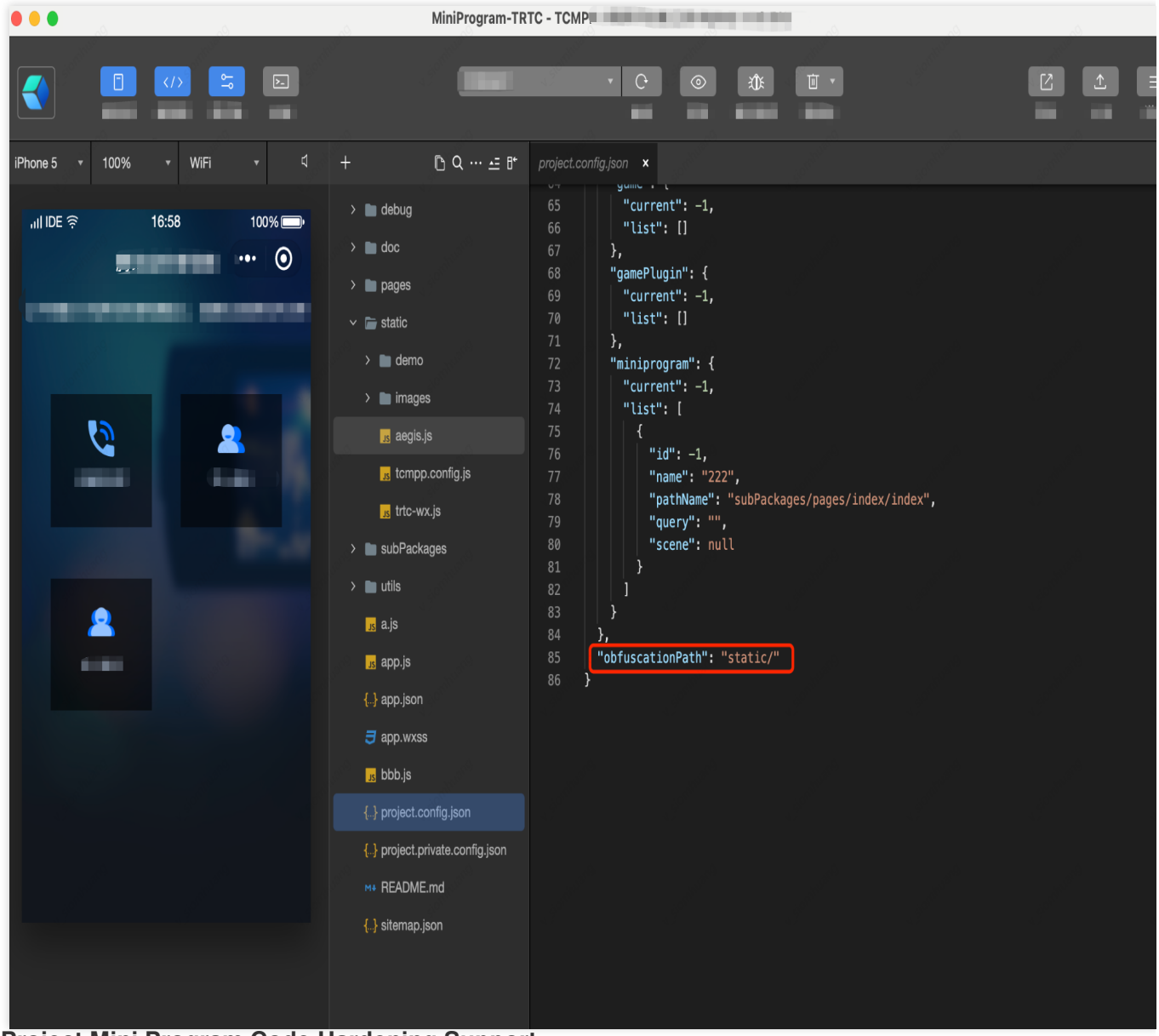
©2013-2025 Tencent Cloud International Pte. Ltd.



2. If you need to modify the location of the tcmpp.config.js file, follow the appropriate steps to ensure it is correctly referenced and encrypted during the build process.If you want to change the directory where tcmpp.config.js file is located, such as putting this file into a static-like directory, you can configure the path of the directory where tcmpp.config.js is located by using the obfuscationPath field in project.config.json. If you don't configure the obfuscationPath field or set it to an empty string, tcmpp.config.js will be found in the same directory as app.json by default.



3. obfuscationPath Configuration The obfuscationPath field is the path to the directory where tcmpp.config.js is located:

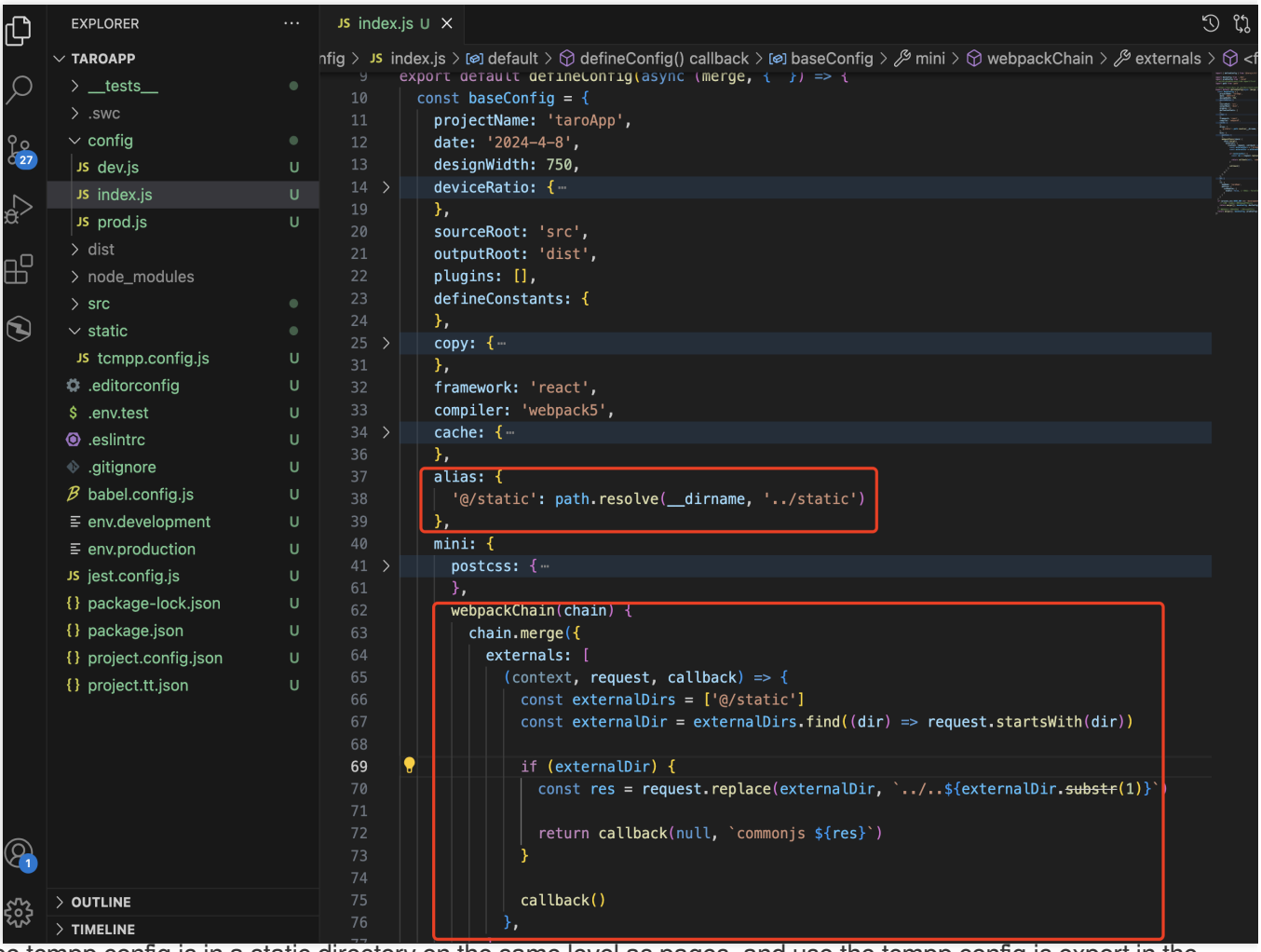


Taro Project Mini Program Code Hardening Support

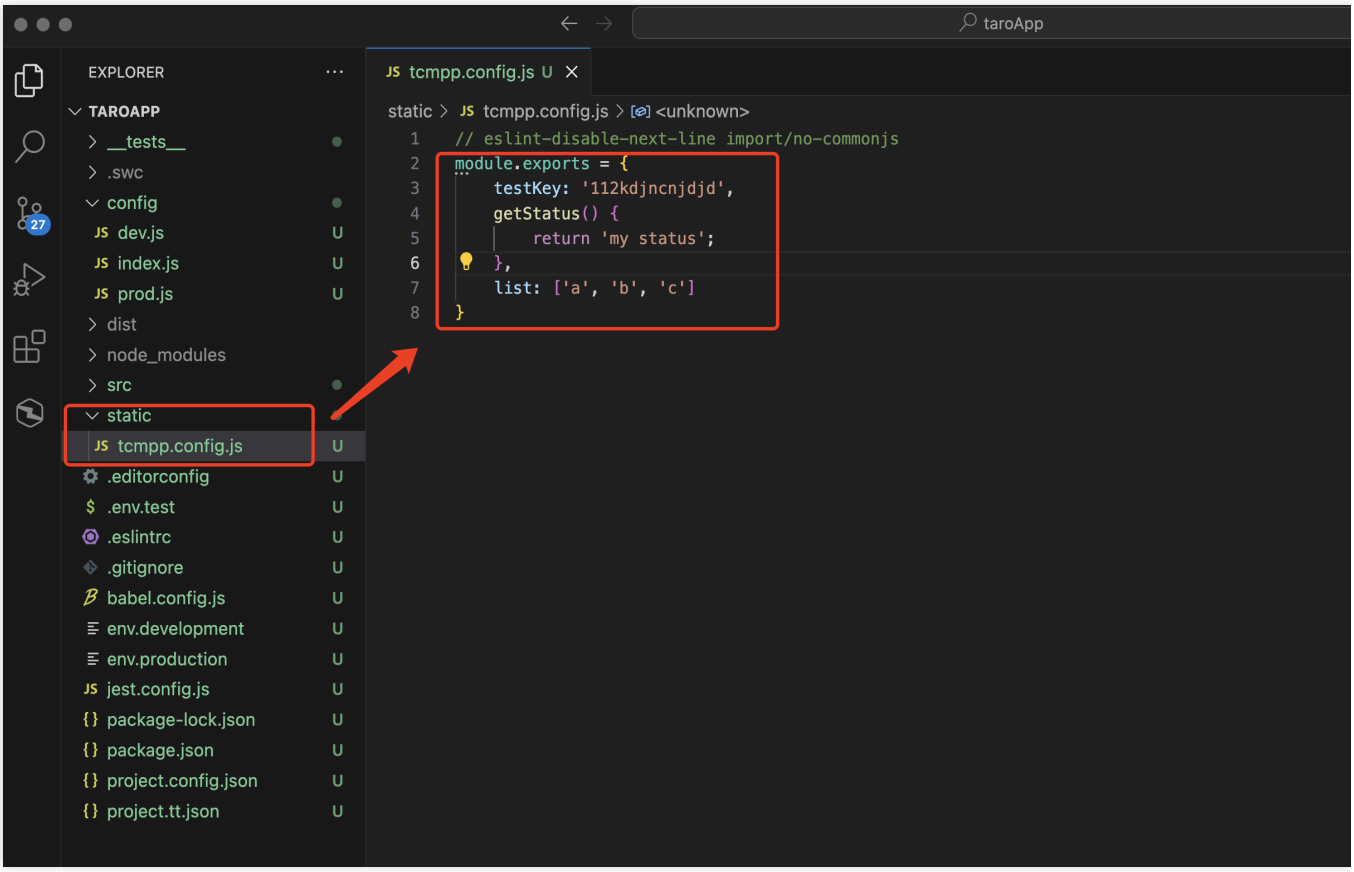
1. In taro mini program, tcmpp.config.js needs to be packaged as an external native public js module, and tcmpp.config.js needs to be written in the same way as mini program js code, i.e., using module.exports to ensure that tcmpp.config.js is not packaged into taro's product but there is a separate tcmpp.config.js file in the dist, so that it can be obfuscated and encrypted in the compilation stage of tcmpp's IDE. After taro is packaged, tcmpp.config.js will not be packaged into the taro product, but a separate tcmpp.config.js file will exist in the dist, so that it can be obfuscated and encrypted during the compilation phase of tcmpp's IDE. Therefore, you need to configure tcmpp.config.js with the externals configured by the taro project. You can see the Taro project's JS file that references the native project.

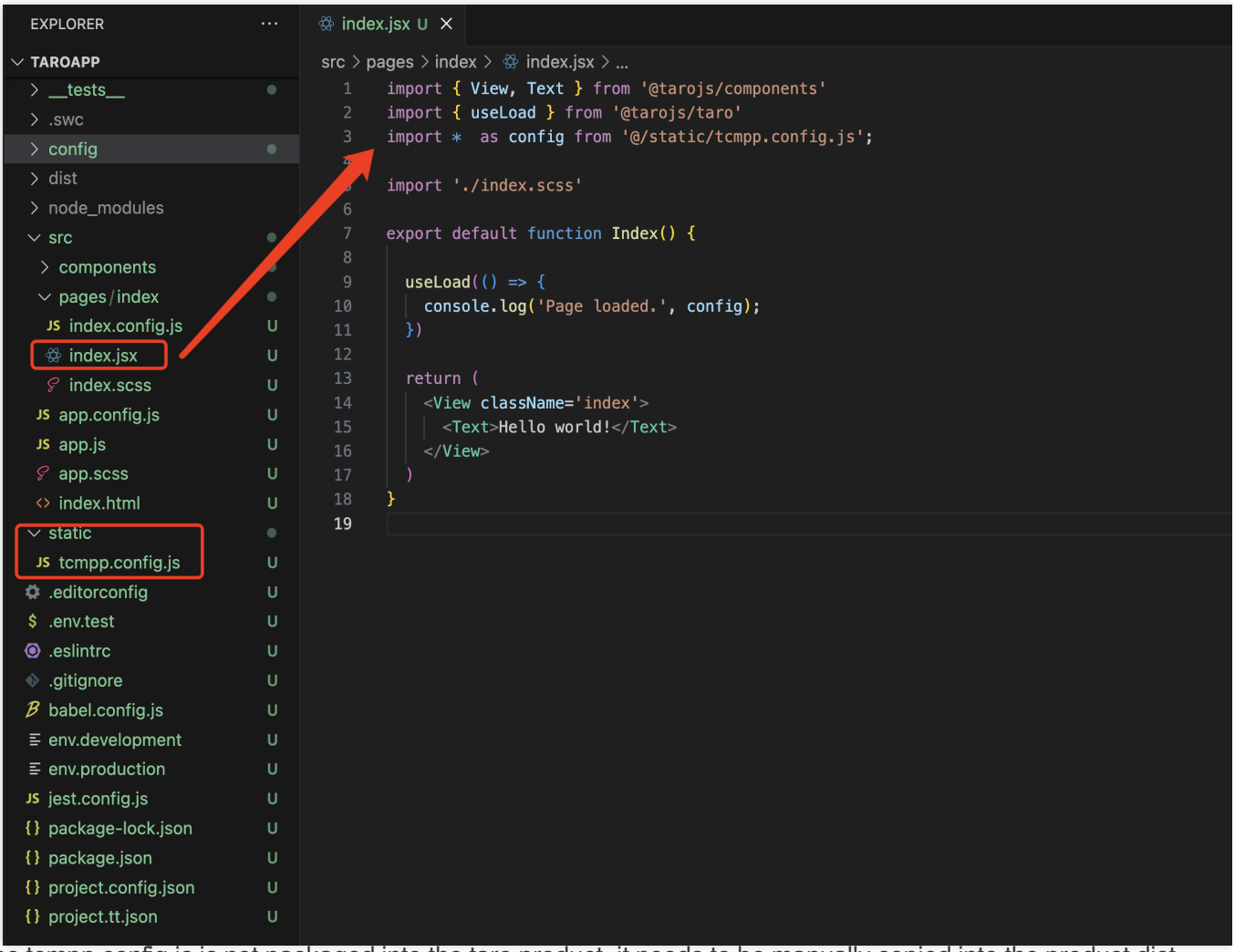
2. Example of a hello world project configuration:

2.1 Configure the taro package, set the static directory to alias, and set the externals configuration:

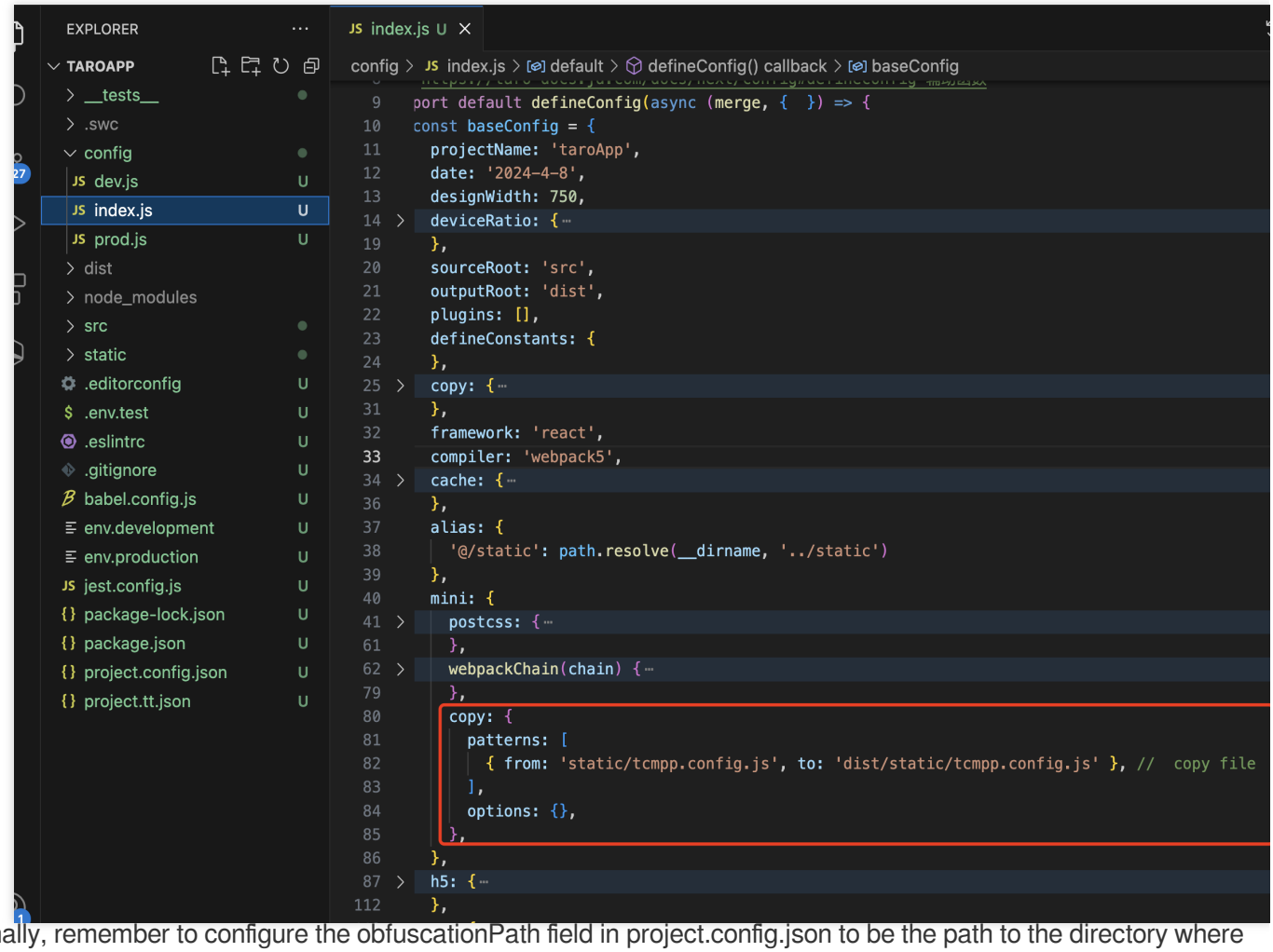


3. Place tcmpp.config.js in a static directory on the same level as pages, and use the tcmpp.config.js export in the project pages/index/index.jsx:

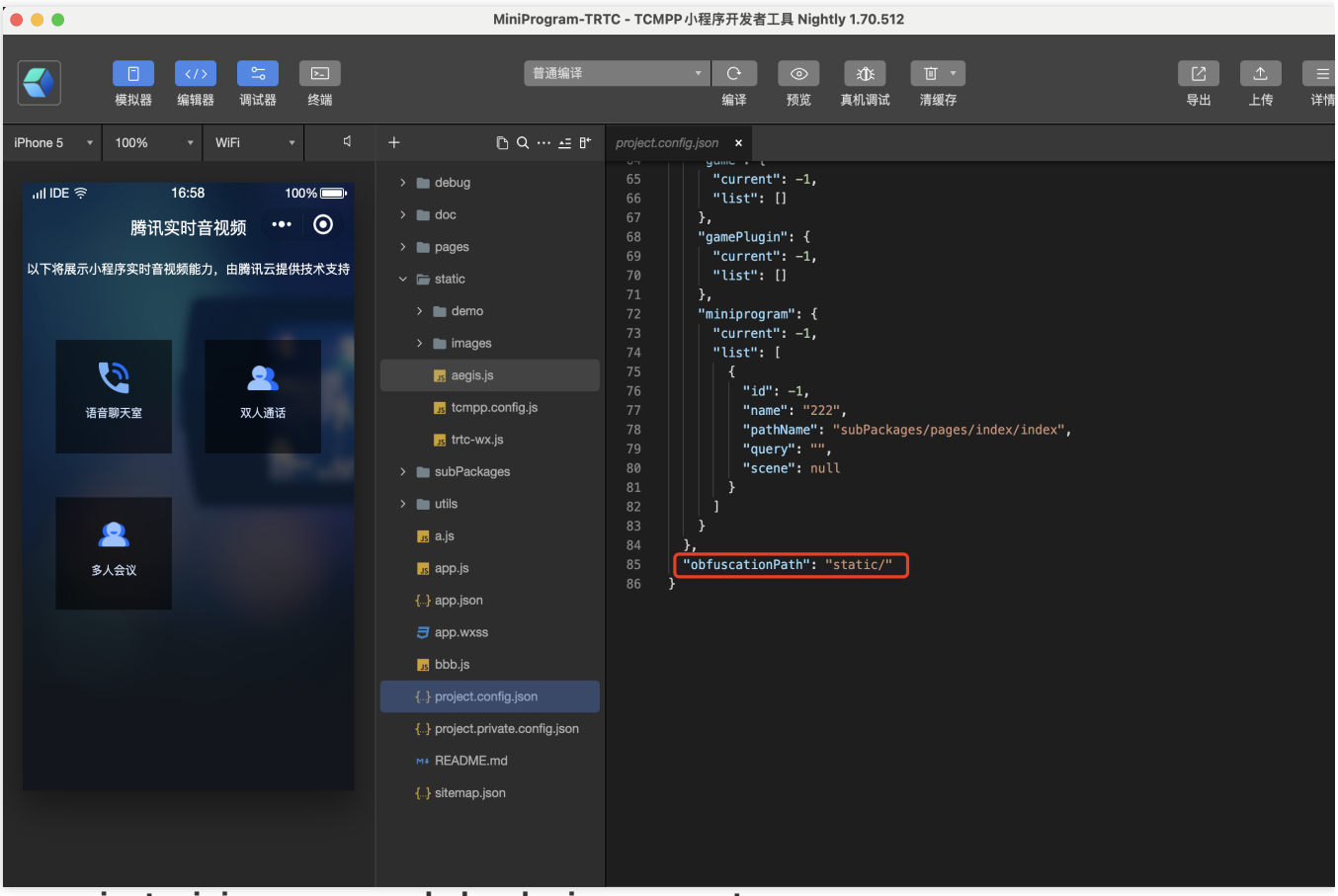

taroApp
EXPLORER
TAROAPP
__tests__
.swc
config
dev.js
index.js
prod.js
dist
node_modules
src
static
tcmpp.config.js
.editorconfig
.env.test
.eslintrc
.gitignore
babel.config.js
env.development
env.production
jest.config.js
package-lock.json
package.json
project.config.json
project.tt.json
tcmpp.config.js U
static > tcmpp.config.js > <unknown>
1 // eslint-disable-next-line import/no-commonjs
2 module.exports = {
3 testKey: '112kdjncnjdjd',
4 getStatus() {
5 return 'my status';
6 },
7 list: ['a', 'b', 'c']
8 }



4. Since tcmpp.config.js is not packaged into the taro product, it needs to be manually copied into the product dist directory via webpack configuration:

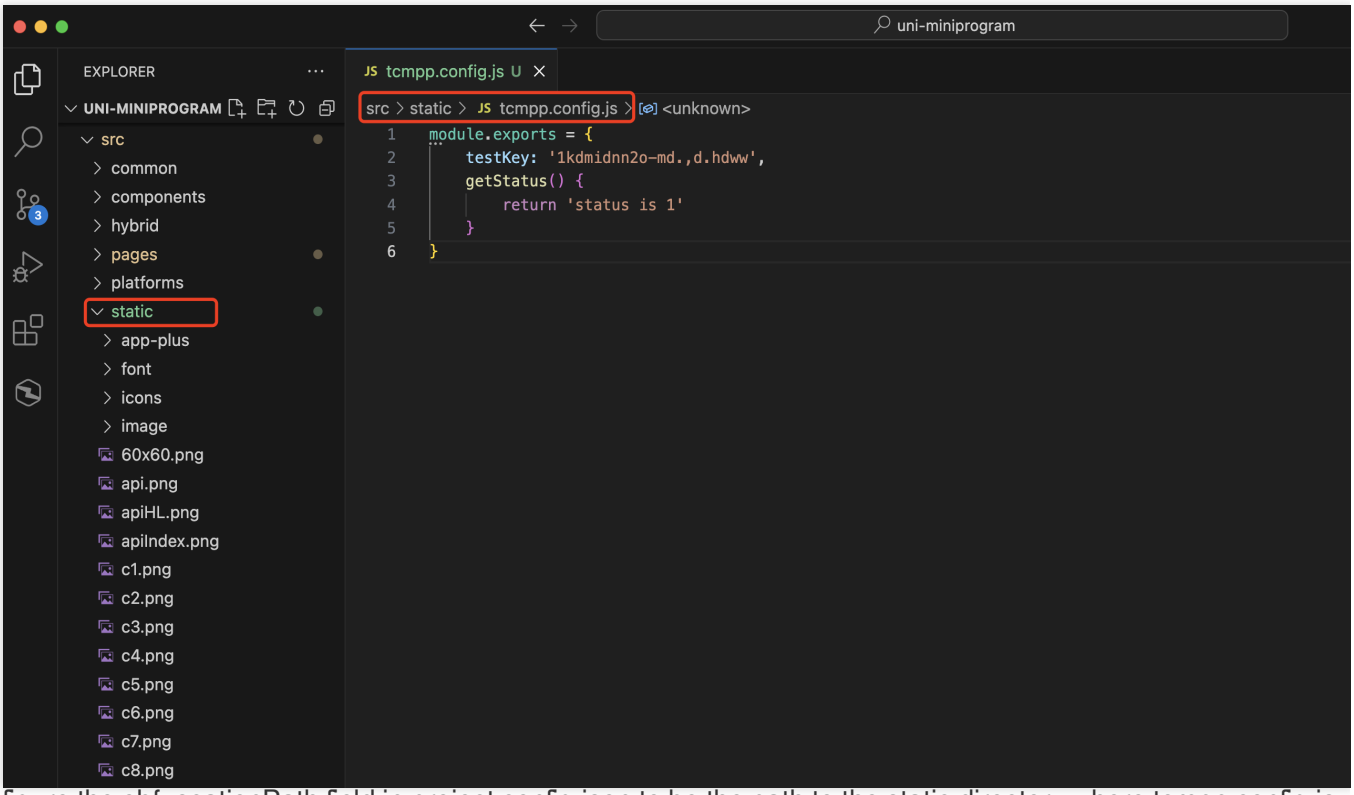


5. Finally, remember to configure the obfuscationPath field in project.config.json to be the path to the directory where tcmpp.config.js is located so that the IDE can find this file for obfuscation.

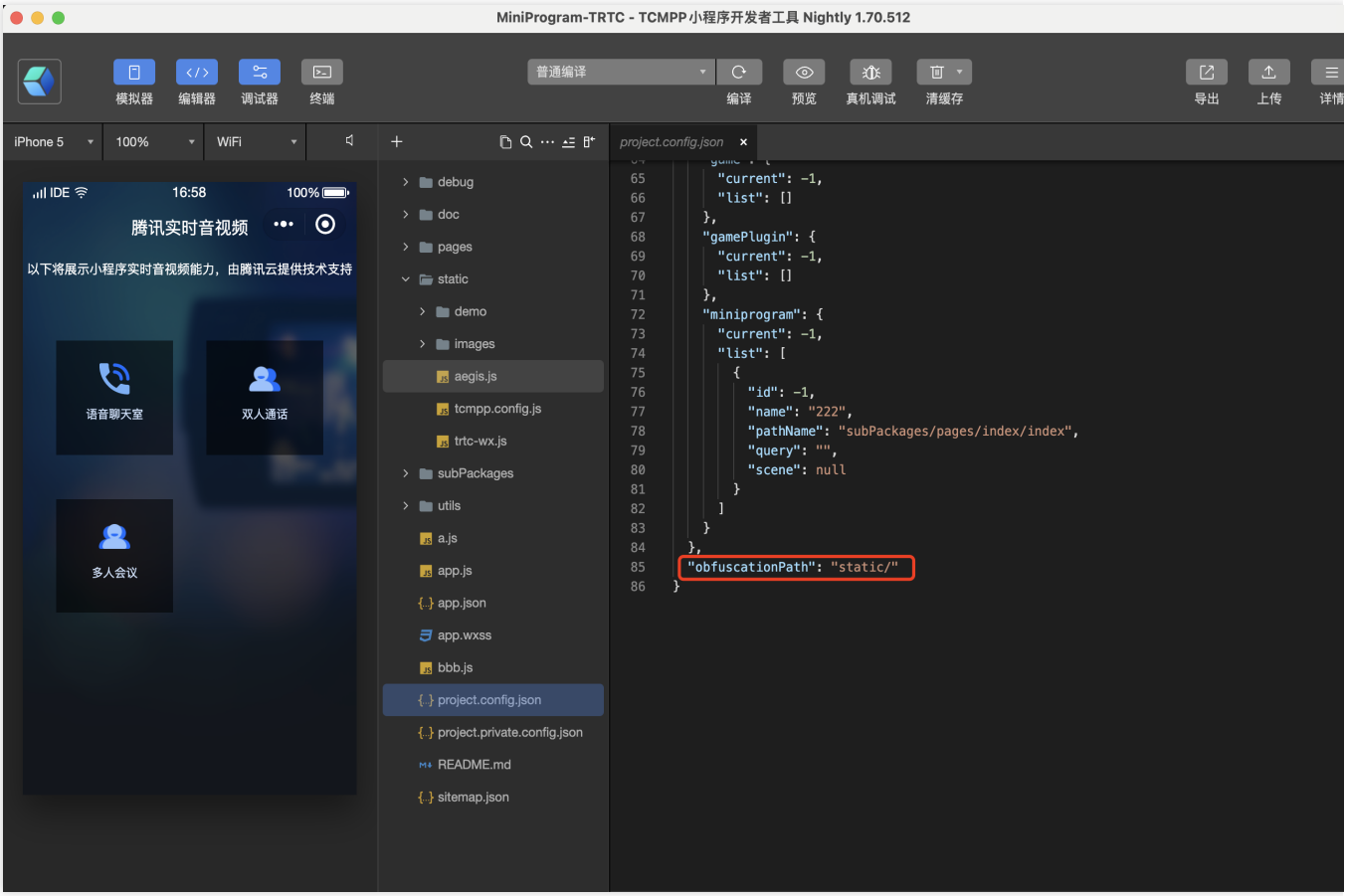


uni-app project mini program code hardening support

1. uni-app provides a static directory, which will be copied to the final package, see uni-app staticuni-app's static directory. Therefore, you need to put tcmpp.config.js into the static directory as well, and use module.exports to export the contents.



2. Configure the obfuscationPath field in project.config.json to be the path to the static directory where tcmpp.config.js is located so that the IDE can find this file for obfuscation.



Using Google Maps in the IDE

Function Introduction

If Google Map-related functions are involved in the business development of the mini program, and you want to implement the preview in the IDE during the development phase, you need to configure the Google JavaScript API key in the project, for example, when using the map component or related APIs:

Components

```
<map type='google' ...></map>
```

API

```
wx.openLocation({
  type: 'google',
  ...
})

wx.chooseLocation({
  type: 'google',
  ...
})
```

Introduction to Use

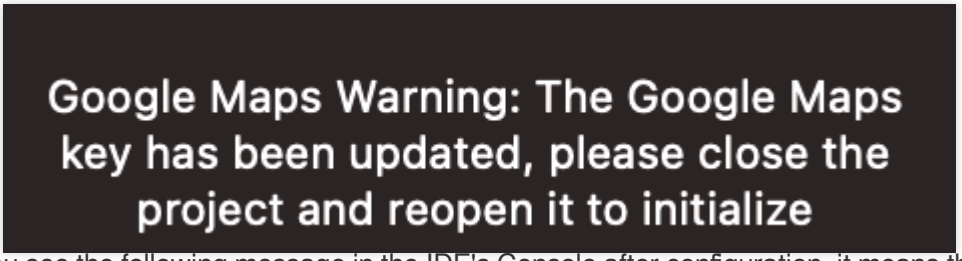
Considering that the user's mini-program business key will use the code hardening capability, in order to facilitate the user's unified management of the key, the Google JavaScript API key needs to be configured in the tcmpp.config.js file, and since the key is only used for IDE preview during the development phase, it is configured using annotations, and it will be configured during the final mini-program compilation and packaging.filtered when the app is finally compiled and packaged.

Usage Process

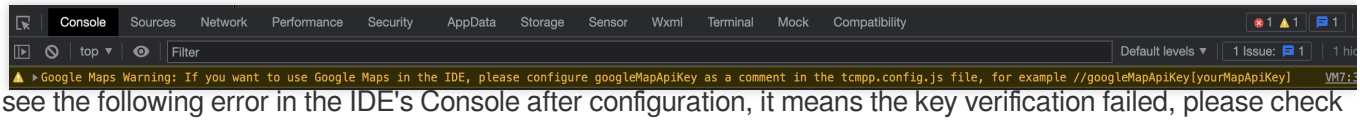
1. The process of creating the tcmpp.config.js file is the same as that of code hardening, please refer to the process of using the code hardening capability of the mini program.
2. Configure the Google JavaScript API key by annotation, the key key in the annotation must be googleMapApiKey and use [] to encapsulate the key, otherwise it will lead to key reading failure, the format is as follows:

```
//googleMapApiKey[xxxxxxxxxxxxxxxxxxxxxxxxxxxx]
```

3. After editing the googleMapApiKey in tcmpp.config.js file in IDE, since you need to re-inject Google JavaScript API script, please close the project and reopen it, IDE will also give you the corresponding pop-up window: Google Maps Warning: The Google Maps key has been updated, please close the project and reopen it to initialize.



4. If you see the following message in the IDE's Console after configuration, it means the configuration of googleMapApiKey has not taken effect, please check whether the configuration follows the relevant format.



If you see the following error in the IDE's Console after configuration, it means the key verification failed, please check if you have configured the correct Google JavaScript API Key.

