relation. If the API started with my.off is called directly, all listening relations will be canceled. Example

copy

```
Page({
  onLoad() {
    this.callback = this.callback.bind(this);
    my.onBLECharacteristicValueChange(this.callback);
  },
  onUnload() {
    // remove listener when page unload
    my.offBLECharacteristicValueChange(this.callback);
  },
  callback(res) {
    console.log(res);
  },
});
```

All other API interfaces accept one object as the parameter. It is possible to specify success (call success), fail (call failure) and complete (call success or failure) cto receive the interface call result. The callback result is generally an object unless otherwise specified. If an error/errorMessage is included, it indicates call failure. The result value after the call is a promise object. Example

copy

```
my.httpRequest({
  url: '/x.htm',
  success(res1) {},
}).then((res2) => {
  // res1 === res2
},(res) => {
  console.log(res.error, res.errorMessage);
})
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_overview

---

Overview {#overview}

Last updated: 2022-07-05

Path: miniprogram_gcash

Overview

2022-07-05 23:31

Introduction

Mini Program extended component library provides important additional capabilities on the basic component library basis. The extended component library contains a set of open-source UI components, which are developed on the basis of Mini Program custom component specifications. Mini Program developers can reuse the extended components rapidly.

Installation

Use the following sample code to install the dependency:

copy

```
$ npm install mini-ali-ui --save
```

Procedures

To use the component, complete the following steps:

1. Register the component in JSON file of the related page. For example, the title component registration is as below:

copy

```
{

  "usingComponents": {

    "title": "mini-ali-ui/es/title/index"
  }
}
```

If you install the rpx version of the extended component library, modify the component name during registration:

copy

```
{

  "usingComponents": {

    "title": "mini-ali-ui-rpx/es/title/index"
  }

}
```

2. Call the component in the AXML file.

copy

```
<title

  hasLine="true"

  iconURL="https://example.com/images/T1HHFgXXVeXXXXXXXX.png"

  type="close"

  onActionTap="titleClose"

>the internal title can be closed
</title>
```

Version upgrade

Upgrade UI component version by using the following command:

copy

```
$ npm update mini-ali-ui --save
```

Note:

The latest version is 1.3.0.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_overview

---

Overview {#overview}

Last updated: 2021-05-09

Path: miniprogram_gcash

Overview

2021-05-09 18:43

Basic Component

The Mini Program provides the developers with a series of basic components so that the developers can combine them for service development.

Attribute Type

The component provides a series of attribute configuration. Each attribute value has the requirement for type:

| | | | | --- | --- | --- | | **Type** | **Description** | **Notes** | | Boolean | Boolean | | | Number | Number | | | String | String | | | Array | Array | | | Object | Object | | | EventHandle | Event handler | Need to define the implementation for the event handler in Page. | | any | Any type | |

Common Component Attribute

All components include the following attributes:

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Description** | **Notes** | | id | String | Unique component identifier. | | | class | String | Style class. | | | style | String | Inline style. | | | data-* | Any | Custom attributes. | When the event is triggered, the custom attribute is transferred to the event handler. | | on *catch* | EventHandle | Event binding, following the ump nomenclature specifications, such as onTap. | Refer to event. |

Tips

The `{{}}` is required to transfer inside the specified attribute type data. For example

copy

```
<view disable-scroll="false"> <!-- Error is a string, not a boolean,
equivalent to boolean type true -->
<view disable-scroll="{{false}}"> <!--right, or empty attribute,
meaning false-->
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/component_overview

---

Overview {#overview}

Last updated: 2022-07-03

Path: miniprogram_gcash

Overview

2022-07-03 18:44

Similar to Page, the customized components consist of four parts: axml, js, json and acss.

There are two steps to create a customized component:

1. Declare the customized component in json. If it is dependent on other components, it is required to declare additionally the dependent customized components.

Sample code:

copy

```
{
  "component": true, // mandate, the value for customized component
must be true
  "usingComponents": {
    "c1":"../x/index"
  }//Dependent component
}
```

Parameter details:

| | | | | | --- | --- | --- | --- | | **Parameter** | **Type** | **Required** | **Description** | | component | Boolean | Yes | Declare customized component. | | usingComponents | Object | No | Path of the customized component in the dependence declaration Absolute project path starts with “/”, and relative path starts with “./” or “../” The npm path does not start with “/”. |

1. Use the Component function to register the customized component. See Component constructor 。

Component parameter description:

| | | | | --- | --- | --- | | **Parameter** | **Description** | **Document** | | onInit | Callback on creation | Component lifecycle. | | deriveDataFromProps | Callback on creation and update | Component lifecycle. | | data | local status | Same as Page (can be modified via setData and $spliceData). | | props | Attribute transferred from outside | Component method and external attribute-props. | | methods | Customized method | Component method and external attribute - methods. |

Sample code:

copy

```
// /components/customer/index.js
Component({
  mixins: [], // minxin easy reuse code
  data: { x: 1 }, // internal data of component
  props: { y: 1 }, // Can add default to attribute transferred from
outside
  onInit() {}, // trigger on component creation, added in version
2.0.0
  deriveDataFromProps(nextProps) {}, // trigger on component creation
and before update, added in version 2.0.0
  didMount(){}, // Lifecycle function
  didUpdate(){},
  didUnmount(){},
  methods: { // customized method
    handleTap() {
      this.setData({ x: this.data.x + 1}); // Can use setData to
```

```
change internal attribute
    },
  },
})
```

in addition, the customized component supports slot and can build flexible page structure. See component template and style .

Sample code:

copy

```
<!-- // /components/customer/index.axml -->
<view>
  <view>x: {{x}}</view>
  <button onTap="handleTap">plusOne</button>
  <slot>
    <view>default slot & default value</view>
  </slot>
</view>
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_custom-component_create-custom-component_overview

---

Overview {#overview}

Last updated: 2022-07-03

Path: miniprogram_gcash

Overview

2022-07-03 18:44

API

The framework provides the developers with more JSAPI and OpenAPI capabilities so that they can launch diversified convenient services to the users.

Notes:

The APIs started with my.on are used to listen to the system events and accept one callback function as the parameter. When the event is triggered, it calls the callback function, which will transfer to the related API started with my.off to cancel the listening relation. If the API started with my.off is called directly, all listening relations will be canceled. Example

copy

```
Page({
  onLoad() {
    this.callback = this.callback.bind(this);
    my.onBLECharacteristicValueChange(this.callback);
  },
  onUnload() {
    // remove listener when page unload
    my.offBLECharacteristicValueChange(this.callback);
  },
  callback(res) {
    console.log(res);
  },
});
```

All other API interfaces accept one object as the parameter. It is possible to specify success (call success), fail (call failure) and complete (call success or failure) cto receive the interface call result. The callback result is generally an object unless otherwise specified. If an error/errorMessage is included, it indicates call failure. The result value after the call is a promise object. Example

copy

```
my.httpRequest({
  url: '/x.htm',
  success(res1) {},
}).then((res2) => {
  // res1 === res2
},(res) => {
  console.log(res.error, res.errorMessage);
})
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_overview

---

Page Configuration {#page-configuration}

Last updated: 2022-07-03

Path: miniprogram_gcash

Page Configuration

2022-07-03 18:44

In the directory `/pages`, the .json file is used to configure the window display of the current page. The page configuration is much simpler than the `app.json` global configuration. It is possible to set the `window` related configuration items only but it is not

necessary to write the `window` key. The page configurations are prior to global configurations.

The `window` configurations are the same as Global Configuration and support the following points additionally:

- Supporting the `optionMenu` configuration navigation icon, which triggers `onOptionMenuClick` on clicking However, please note that the `optionMenu` configuration will be deprecated.
- Supporting `titlePenetrate` to set the navigation bar click-through
- Supporting `barButtonTheme` to set the navigation bar icon scheme

Complete configurations

| | | | | | --- | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | **Minimum version** | | optionMenu | Object | NO | Set extra icon of navigation bar, supporting attribute icon with value as icon URL (starting with https/http) or base64 string, suggested size 30*30 px. | Base library 1.3.0 | | titlePenetrate | BOOL | NO | Set navigation bar click-through. | - | | barButtonTheme | String | NO | Set navigation bar icon scheme, “default” for blue icon or “light” for white icon. | - |

Here is a basic example:

copy

```
{
    "optionMenu": {
    "icon": "https://img.example.com/example.png"
    },
    "titlePenetrate": "YES",
    "barButtonTheme": "light"
}
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_page_page-configuration

---

Page Configuration {#page-configuration}

Last updated: 2021-05-09

Path: miniprogram_gcash

Page Configuration

2021-05-09 18:43

In the directory `/pages`, the .json file is used to configure the window display of the current page. The page configuration is much simpler than the `app.json` global configuration. It is possible to set the `window` related configuration items only but it is not necessary to write the `window` key. The page configurations are prior to global configurations.

The `window` configurations are the same as Global Configuration and support the following points additionally:

- Supporting the `optionMenu` configuration navigation icon, which triggers `onOptionMenuClick` on clicking However, please note that the `optionMenu` configuration will be deprecated.
- Supporting `titlePenetrate` to set the navigation bar click-through
- Supporting `barButtonTheme` to set the navigation bar icon scheme

Complete configurations

| | | | | | --- | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | **Minimum version** | | optionMenu | Object | NO | Set extra icon of navigation bar, supporting attribute icon with value as icon URL (starting with https/http) or base64 string, suggested size 30*30 px. | Base library 1.3.0 | | titlePenetrate | BOOL | NO | Set navigation bar click-through. | - | | barButtonTheme | String | NO | Set navigation bar icon scheme, “default” for blue icon or “light” for white icon. | - |

Here is a basic example:

copy

```
{
    "optionMenu": {
    "icon": "https://img.example.com/example.png"
    },
    "titlePenetrate": "YES",
    "barButtonTheme": "light"
}
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_page_page-configuration

---

Page FAQ {#page-faq}

Last updated: 2021-05-09

Path: miniprogram_gcash

Page FAQ

2021-05-09 18:43

Q: How to use cookie in Mini Program

A: Cookie is not suggested to be used in Mini Program, the cookie set by server side will not be forbidden and it will be set to the Mini Program process. In the next request, the cookie will be set into the request automatically. In front side, cookie can not be obtained and will do nothing about the cookie.

Q: How to get the parameters of onload in certain page

A: From `getCurrentPages`, the instance of page stack can be obtained, then parameters can be obtained.

Q: Can Mini Program able to listen to the close event? Which function will be invoked when clicking the close button?

A: Mini Program can not listen to the close event and nothing will be invoked when clicking the close button.

Q: Data does not refresh when calling setData

A: Please check the effectiveness of `this` instance and make sure the code logic is correct.

Q: Blank page displays when jumping to a new page, how to solve it

A: If using `request` to get data from server side, please make sure the domain whitelist is configured. If not, the data can not be requested, which may causing blank page.

Q: How to get the parameter in the link when jumping to a new page

A: Use `onLaunch` to listen to the initialization of Mini Program, the query can be obtained from the `onLaunch` parameter.

Q: How to import js in Mini Program

A: Use `import {Ajax} from '/util(or ./util)'` to import js

Q: How to trigger a function automatically without clicking

A: Call the function in `onload` or `onshow`.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_page_page-faq

---

Page FAQ {#page-faq}

Last updated: 2022-07-03

Path: miniprogram_gcash

Page FAQ

2022-07-03 18:44

Q: How to use cookie in Mini Program

A: Cookie is not suggested to be used in Mini Program, the cookie set by server side will not be forbidden and it will be set to the Mini Program process. In the next request, the cookie will be set into the request automatically. In front side, cookie can not be obtained and will do nothing about the cookie.

Q: How to get the parameters of onload in certain page

A: From `getCurrentPages`, the instance of page stack can be obtained, then parameters can be obtained.

Q: Can Mini Program able to listen to the close event? Which function will be invoked when clicking the close button?

A: Mini Program can not listen to the close event and nothing will be invoked when clicking the close button.

Q: Data does not refresh when calling setData

A: Please check the effectiveness of `this` instance and make sure the code logic is correct.

Q: Blank page displays when jumping to a new page, how to solve it

A: If using `request` to get data from server side, please make sure the domain whitelist is configured. If not, the data can not be requested, which may causing blank page.

Q: How to get the parameter in the link when jumping to a new page

A: Use `onLaunch` to listen to the initialization of Mini Program, the query can be obtained from the `onLaunch` parameter.

Q: How to import js in Mini Program

A: Use `import {Ajax} from '/util(or ./util)'` to import js

Q: How to trigger a function automatically without clicking

A: Call the function in `onload` or `onshow`.

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_page_page-faq

---

Page Introduction {#page-introduction}

Last updated: 2021-05-09

Path: miniprogram_gcash

Page Introduction

2021-05-09 18:43

Page represents a page of Mini Program, taking charge of the display and interaction of page. Each page will have a subdirectory in the project, basically there are as many subdirectories as there are pages. It is also a constructor to generate instance of page.

Each page consists of four files:

- `[PageName].js`: page logic
- `[PageName].axml`: page structure
- `[PageName].json`: page configuration (optional)
- `[PageName].acss`: page style sheet (optional)

When page is initialized, the data should be provided.

copy

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

According to the data provided, the page can be rendered.

copy

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

The function should be specified when defining interaction.

copy

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

The above code shows when user clicks the view, the `handleTap` function will be invoked.

copy

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

If you want to re-render the page, you need to call `this.setData` function in the `[PageName].js` script.

copy

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

The above code shows when user click the view, the `changeText` function will be invoked.

copy

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

In the `changeText` function, `this.setData` is called to change the text data, and then the page will re-render to show the changed data.

九色鹿

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_page_overview

---

Page Introduction {#page-introduction}

Last updated: 2022-07-03

Path: miniprogram_gcash

Page Introduction

2022-07-03 18:44

Page represents a page of Mini Program, taking charge of the display and interaction of page. Each page will have a subdirectory in the project, basically there are as many subdirectories as there are pages. It is also a constructor to generate instance of page.

Each page consists of four files:

- `[PageName].js`: page logic

- `[PageName].axml`: page structure
- `[PageName].json`: page configuration (optional)
- `[PageName].acss`: page style sheet (optional)

When page is initialized, the data should be provided.

copy

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

According to the data provided, the page can be rendered.

copy

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

The function should be specified when defining interaction.

copy

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

The above code shows when user clicks the view, the `handleTap` function will be invoked.

copy

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

If you want to re-render the page, you need to call `this.setData` function in the `[PageName].js` script.

copy

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

The above code shows when user click the view, the `changeText` function will be invoked.

copy

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

In the `changeText` function, `this.setData` is called to change the text data, and then the page will re-render to show the changed data.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_page_overview

---

Page Mechanism {#page-mechanism}

Last updated: 2022-06-30

Path: miniprogram_gcash

Page Mechanism

2022-06-30 23:35

Page(object: Object)

Each .js file in the `/pages` directory has a `Page` object to define properties for a Mini Program page. We can use this object to specify the initial data, register lifecycle callbacks, and customize event handlers.

Below are the basic page codes:

copy

```
// pages/index/index.js
Page({
  data: {
    title: "Mini Program",
  },
  onLoad(query) {
    // Page loading
  },
  onShow() {
```

```
    // Page showing
  },
  onReady() {
    // Page loading complete
  },
  onHide() {
    // Page hiding
  },
  onUnload() {
    // Page closed
  },
  onTitleClick() {
    // Title clicked
  },
  onPullDownRefresh() {
    // Page pulled down
  },
  onReachBottom() {
    // Page pulled down till bottom
  },
  onShareAppMessage() {
   // Return customized sharing information
  },
  // Event handler object
  events: {
    onBack() {
      console.log('onBack');
    },
  },
  // Custom event handler
  viewTap() {
    this.setData({
      text: 'Set data for update.',
    });
  },
  // Custom event handler
  go() {
    // Jump with parameters, read type from query of onLoad function
in page/ui/index
    my.navigateTo({url:'/page/ui/index?type=mini'});
  },
  // Custom data object
  customData: {
    name: 'Mini Program',
  },
});
```

Page Lifecycle

The diagram below shows the lifecyle of the `Page` object.

The Mini Program basically uses the view thread (Webview) and application service thread (Worker) for control and management. The Webview and Worker threads run in parallel.

- Upon startup, the Worker thread invokes `app.onLauch` and `app.onShow` when the app is created. Subsequently when WebView initialization completes, the Worker thread receives a notification from WebView thread and then invokes `page.onLoad` and `page.onShow` to indicate the completion of page creation.

- Upon the notification on completion of the Webview initialization, the Worker sends the initialized data to the Webview for render. Now the Webview completes the first data render.

- After the first render is completed, the Webview enters into the ready status and notifies the Worker. The Worker calls the `page.onReady` function and enters into the active status.

- in the active status, the Worker modifies data each time and then notifies the Webview for rendering. When switched to the background, the Worker calls the `page.onHide` function and enters into the suspended status. The `page.onShow` function will be called when page returns to the foreground and enters into the active status. When the return or redirection page is called, the function `page.onUnload` is called for page destroying.

Object Attribute Description

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Description** | **Minimum version** | | data | Object | Function | Function for initializing data or returning initialized data. | - | | events | Object | Event handler object. | 1.13.7 | | onLoad | Function(query: Object) | Trigger on page loading. | - | | onShow | Function | Trigger on page showing. | - | | onReady | Function | Trigger on completion of initial page rendering. | - | | onHide | Function | Trigger on page hiding. | - | | onUnload | Function | Trigger on page unloading. | - | | onShareAppMessage | Function(options: Object) | Trigger on clicking upper-right corner share. | - | | onTitleClick | Function | Trigger on clicking title. | - | | onOptionMenuClick | Function | Trigger on clicking extra icon of navigation bar. | 1.3.0 | | onPopMenuClick | Function | Trigger on clicking custom menu buttons in upper-right general menu. | 1.3.0 | | onPullDownRefresh | Function({from: `manual` | `code`}) | Trigger on pulling down page. | - | | onPullIntercept | Function | Trigger on pulling down interruption. | 1.11.0 | | onTabItemTap | Function | Trigger on clicking `tabItem`. | 1.11.0 | | onPageScroll | Function({scrollTop}) | Trigger on page scrolling. | - | | onReachBottom | Function | Trigger on pulling page till bottom. | - | | Others | Any | The developer can add any function or attribute column into the `object`. The `this` can be used for access in the page functions. | - |

Page Data Object

The initial `data` can be specified for the page by setting data. When data is an object, it is shared by all pages. In other words, when it returns and then enters the page again, the last page data will be displayed instead of the initial `data`. In such a case, the issue may be fixed by setting data as unchanged data or changing data as page exclusive `data`.

Set as unchanged data

copy

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

Set as page exclusive data (not recommended)

copy

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

Notes:

Do not modify `this.data` directly, which will not change the page status and will cause data inconsistency.

For example:

copy

```
Page({
 data: { arr:[] },
 doIt() {
   this.data.arr.push(1); // Do not do this!
   this.setData({arr: this.data.arr});
 }
});
```

Lifecycle Function

onLoad(query: Object)

Trigger on page initializing. It called only once for each page.

The query is the query object transferred in the `my.navigateTo` and `my.redirectTo`.

The query content is in the format: "parameter name=parameter value&parameter name=parameter value…"

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | query | Object | Parameter for opening the current page path. |

onShow()

Trigger on page showing or switching to foreground

onReady()

Trigger on completion of initial page rendering. It is called only once for each page, indicating the page is ready and can interact with view layer. For the setting of interface such as `my.setNavigationBar`, please set behind `onReady`.

onHide()

Trigger on page hiding or switching to background. Such as `my.navigateTo` to another page or switching via bottom tab.

onUnload()

Trigger on page unloading. Such as `my.redirectTo` or `my.navigateBack` to another page.

Page Event Handler

onShareAppMessage(options: Object)

Trigger on clicking the **Share** button in upper-right general menu or clicking in-page **Share** button.

Define the `onShareAppMessage` function in `Page` and set the sharing information:

- Display the **Share** button in the upper-right menu of every page by default. Only the shared content can be customized by using the `onShareAppMessage` function.
- The `onShareAppMessage` function is called when the user clicks the **Share** button.
- This event handler must return an Object to customize the shared content.
- The mini program supports to trigger the sharing by using the button component. The value of `open-type` is `share`.

Sample codes:

copy

```
// API-DEMO page/API/share/share.json
 {
        "defaultTitle" : "onShareAppMessage"
 }
```

copy

```
 <view class = "page" >
 <view class = "page-description" > Click the upper-right menu to
```

```
customize the sharing </view>
 </view>
```

copy

```
// API-DEMO page/API/share/share.js
 Page ({
  onShareAppMessage () {
  return {
    title : 'Sharing the View component' ,
    desc : 'The View component is general' ,
    path : 'page/component/view/view' ,
    };
  },
 });
```

Sample codes for triggering the sharing with the button component:

copy

```
<view>
<button type="primary" open-type="share" a:if="
{{canIUseShareButton}}">Share to friends</button>
</view>
```

copy

```
Page({
  data: { canIUseShareButton: true },
  setShareButtonSwitch () { this.setData({ canIUseShareButton:
my.canIUse('button.open-type.share') }) },
    onLoad() { this.setShareButtonSwitch(); } ,

    onShareAppMessage() {
    return {
      title: 'Mini program demo',
      desc: 'Mini program official demo that displays the supported
APIs and components',
      path: 'page/component/component-pages/view/view?param=123'
    }
  }
});
```

The parameters are in Object type and have the following attributes:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | from | String | Source of triggering sharing event. Valid values are:
- `button`: click the button in the page to trigger the sharing;
- `menu`: click the button in the upper-right menu to trigger the sharing;
- `code`: call the my.showSharePanel API to trigger the sharing. | | target | Object | If the value of `from` is `button`, `target` is the `button` that triggers the event. Otherwise, `button` is `undefined`. | | webViewUrl | String | When the page contains the web-view component, return the URL of the current web-view. |

This event handler must return an Object to customize the shared content.

Return value

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | title | String | Yes | Customized sharing title. Max 50 characters. | | desc | String | No | Customized description about the sharing. The maximum length is 140 characters when sharing to Sina Weibo, so it's suggested that the description does not exceed 140 characters. | | path | String | Yes | Customized sharing path. The customized parameters in the path can be obtained from the `onLoad` lifecycle function and follow the HTTP GET rules. The path cannot contain the root directory (/). | | imageUrl | String | No | The path of the customized icon, which can be a web image path. Recommended Image size is 1200 x 630 pixels and should not be more than 8M, and Minimum image size is 200 x 200 pixels. | | bgImgUrl | String | No | The path of the customized image, which can be a web image path. The image size is suggested to be 750 x 825 pixels. | | success | Function | No | The callback method that indicates a successful sharing. | | fail | Function | No | The callback method that indicates a failed sharing. |

Success callback function

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | channelName | String | The sharing channel. | | shareResult | Boolean | The result that indicates whether the sharing is successful. |

onTitleClick()

Trigger on clicking title.

onOptionMenuClick()

Trigger on clicking upper-right corner menu button.

onPopMenuClick()

Trigger on clicking upper-right corner general menu button.

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

Trigger on pulling down to refresh. It is required to enable `pullRefresh` in the `window` option of app.json. When the data refresh is processed completely, call `my.stopPullDownRefresh` to stop the pull-to-refresh for that page.

onPullIntercept()

Trigger on pulling down interruption.

onTabItemTap(object: Object)

Trigger on clicking `tabItem`

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | from | String | Click source. | | pagePath | String | Page path of the clicked tabItem. | | text | String | Button text of the clicked tabItem. | | index | Number | Number of the clicked tabItem, starting from 0. |

onPageScroll({scrollTop})

Trigger on page scrolling, `scrollTop` is the page scrolling distance.

onReachBottom()

Trigger on pulling page till bottom.

Events

To simplify codes, a new `event` handler object events is available. The existing page handler is equivalent to the exposed event functions on the page instance.

Notes:

- The support for events starts from basic library version 1.13.7.
- Please distinguish the basic library version requirements for the same named functions of the page event handler and events.

Below is the list of event functions supported by events:

| | | | | | --- | --- | --- | --- | | **Event** | **Type** | **Description** | **Lowest version** | | onBack | Function | Trigger on page returning. | 1.13.7 | | onKeyboardHeight | Function | Trigger on keyboard height changing. | 1.13.7 | | onOptionMenuClick | Function | Trigger on clicking upper-right corner menu button. | 1.13.7 | | onPopMenuClick | Function | Trigger on clicking upper-right corner general menu button. | 1.13.7 | | onPullIntercept | Function | Trigger on pulling down interruption. | 1.13.7 | | onPullDownRefresh | Function({from: manual/code}) | Trigger on pulling down page. | 1.13.7 | | onTitleClick | Function | Trigger on clicking title. | 1.13.7 | | onTabItemTap | Function | Trigger on click non-current tabItem. | 1.13.7 | | beforeTabItemTap | Function | Trigger before click non-current tabItem. | 1.13.7 | | onResize | Function({size: {windowWidth: number, windowHeight: number}}) | Trigger on window size changing. | 1.16.0 |

Sample code:

copy

```
// Feature detection
my.canIUse('page.events.onBack');

Page({
  data: {
    text: 'This is page data.'
  },
  onLoad(){
    // trigger on page loading
  },
```

```
  events:{
    onBack(){
      // Trigger on page returning
    },
    onKeyboardHeight(e){
      // Trigger on keyboard height changing
      console.log('keyboard height:', e.height)
    },
    onOptionMenuClick(){
      // Trigger on clicking upper-right corner menu button
    },
    onPopMenuClick(e){
      // Trigger on clicking custom menu buttons in upper-right
general menu
      console.log('index of the clicked custom menu', e.index)
      console.log('name of the clicked custom menu', e.name)
      console.log('menuIconUrl of the clicked custom menu',
e.menuIconUrl)
    },
    onPullIntercept(){
      // Trigger on pulling down interruption
    },
    onPullDownRefresh(e){
      // Trigger on pulling down page The e.from value “code”
indicates the event triggered by startPullDownRefresh; value “manual”
indicates the pull-down event trigger by user
      console.log('type of triggered pull-down refresh', e.from)
      my.stopPullDownRefresh()
    },
    onTitleClick(){
      // Trigger on clicking title
    },
    onTabItemTap(e){
      // e.from means triggering after clicking tabItem and switching;
value “user” indicates event triggered by user clicking; value “api”
indicates event triggered by switchTab
      console.log('type of triggering tab change', e.from)
      console.log('path of page corresponding to the clicked tab',
e.pagePath)
      console.log('text of the clicked tab', e.text)
      console.log('index of the clicked tab', e.index)
    },
    beforeTabItemTap(){
      // trigger on clicking tabItem but before switching
    },
    onResize(e){
      // Trigger on window size changing
      var {windowWidth, windowHeight} = e.size
      console.log('width of changed window', windowWidth)
      console.log('height of changed window', windowHeight)
    },
```

```
  }
})
```

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

The `setData` sends data from logic layer to view layer and changes the value of `this.data`.

The `Object` is expressed in the form `key: Value`.. The `key` value in `this.data` is changed to `value`. Here, the `key` can be flexibly provided in form of data path, such as `array[2].message`, `a.b.c.d`. It is not necessary to predefine in `this.data`.

The following points are worth attentions in use:

1. It is invalid to modify `this.data` directly, which will not change the page status and will cause data inconsistency.
2. Only the JSON supported data is supported.
3. Try not to set too many data once.
4. Do not set any value in the data as undefined, otherwise, that item will not be set, and potential issue may arise.

Sample code:

copy

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

copy

```
Page({
  data: {
    text: 'test',
    array: [{text: 'a'}],
    object: {
      text: 'blue',
    },
    name: 'Mini Program',
  },
  changeTitle() {
```

```
    // Wrong! Do not modify the data directly
    // this.data.text = 'changed data'

    // Correct!
    this.setData({
      text: 'ha',
    });
  },
  changeArray() {
    // Possible to modify data by using directly data path
    this.setData({
      'array[0].text': 'b',
    });
  },
  changePlanetColor(){
    this.setData({
      'object.text': 'red',
    });
  },
  addNewKey() {
    this.setData({
      'newField.text': 'c',
    });
  },
  changeName() {
    this.setData({
      name: 'Mini Program',
    }, () => { // Accept transfer of callback function
      console.log(this); // this: current page instance
      this.setData({ name: this.data.name + ', ' + 'welcome!'});
    });
  },
});
```

Parameter description:

| | | | | --- | --- | --- | --- | | **Event** | **Type** | **Description** | **Lowest version** | | data | Object | Data to be changed. | - | | callback | Function | Callback function, to be executed on completion of page rendering and update. | 1.7.0, Use `my.canIUse('page.setData.callback')` for compatibility processing. |

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

Note: `$spliceData` is supported since version 1.7.2. The **my.canIUse('page.$spliceData')** can be used for compatibility processing.

Similarly, the `spliceData` is used to transfer data from logic layer to view layer, but has higher performance than `setData` in processing long list.

The `Object` is expressed in the form `key: Value`.. The `key` value in `this.data` is changed to `value`. Here, the key can be flexibly provided in form of data path, such as `array[2].message`, `a.b.c.d`. It is not necessary to predefine in `this.data`. The `value` is an array (format: [start, deleteCount, ...items]). The first element of the array is the start position of the operation, the second element is the number of elements to be deleted, and other other elements are the insertion data. It maps the array `splice` method in `es5`.

Sample code:

copy

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

copy

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

Page output:

copy

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

Parameter description:

| | | | --- | --- | --- | | **Event** | **Type** | **Description** | | data | Object | Data to be changed. | | callback | Function | Callback function, to be executed on completion of page rendering and update. |

Page.prototype.$batchedUpdates(callback: Function)

Batch update data.

Note: `$batchedUpdates` is supported since version 1.14.0. Themy.canIUse('page.$batchedUpdates') can be used for compatibility processing.

Parameter description:

| | | | | --- | --- | --- | | **Event** | **Type** | **Description** | | callback | Function | The data operation in the callback function will be updated in batch. |

Sample code:

copy

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

copy

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

1. In this example, page `counter` adds 2 on each button clicking.
2. The `setData` is placed within this.$batchedUpdates. Thus, only one data transfer happens despite of multiple `setData`.

Page.route

Path of `Page`, mapping the path value configured in app.json, type `String`

> This is a read-only attribute.

copy

```
Page({
  onShow() {
    // Map the path value configured in app.json
    console.log(this.route)
  }
})
```

九色鹿

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_page_page-mechanism

---

Page Mechanism {#page-mechanism}

Last updated: 2022-07-03

Path: miniprogram_gcash

Page Mechanism

2022-07-03 18:44

Page(object: Object)

Each .js file in the `/pages` directory has a `Page` object to define properties for a Mini Program page. We can use this object to specify the initial data, register lifecycle callbacks, and customize event handlers.

Below are the basic page codes:

copy

```
// pages/index/index.js
Page({
  data: {
    title: "Mini Program",
  },
  onLoad(query) {
    // Page loading
  },
  onShow() {
    // Page showing
  },
  onReady() {
    // Page loading complete
```

```
  },
  onHide() {
    // Page hiding
  },
  onUnload() {
    // Page closed
  },
  onTitleClick() {
    // Title clicked
  },
  onPullDownRefresh() {
    // Page pulled down
  },
  onReachBottom() {
    // Page pulled down till bottom
  },
  onShareAppMessage() {
   // Return customized sharing information
  },
  // Event handler object
  events: {
    onBack() {
      console.log('onBack');
    },
  },
  // Custom event handler
  viewTap() {
    this.setData({
      text: 'Set data for update.',
    });
  },
  // Custom event handler
  go() {
    // Jump with parameters, read type from query of onLoad function
in page/ui/index
    my.navigateTo({url:'/page/ui/index?type=mini'});
  },
  // Custom data object
  customData: {
    name: 'Mini Program',
  },
});
```

Page Lifecycle

The diagram below shows the lifecyle of the `Page` object.

The Mini Program basically uses the view thread (Webview) and application service thread (Worker) for control and management. The Webview and Worker threads run in parallel.

- Upon startup, the Worker thread invokes `app.onLauch` and `app.onShow` when the app is created. Subsequently when WebView initialization completes, the Worker thread receives a notification from WebView thread and then invokes `page.onLoad` and `page.onShow` to indicate the completion of page creation.

- Upon the notification on completion of the Webview initialization, the Worker sends the initialized data to the Webview for render. Now the Webview completes the first data render.

- After the first render is completed, the Webview enters into the ready status and notifies the Worker. The Worker calls the `page.onReady` function and enters into the active status.

- in the active status, the Worker modifies data each time and then notifies the Webview for rendering. When switched to the background, the Worker calls the `page.onHide` function and enters into the suspended status. The `page.onShow` function will be called when page returns to the foreground and enters into the active status. When the return or redirection page is called, the function `page.onUnload` is called for page destroying.

Object Attribute Description

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Description** | **Minimum version** | | data | Object | Function | Function for initializing data or returning initialized data. | - | | events | Object | Event handler object. | 1.13.7 | | onLoad | Function(query: Object) | Trigger on page loading. | - | | onShow | Function | Trigger on page showing. | - | | onReady | Function | Trigger on completion of initial page rendering. | - | | onHide | Function | Trigger on page hiding. | - | | onUnload | Function | Trigger on page unloading. | - | | onShareAppMessage | Function(options: Object) | Trigger on clicking upper-right corner share. | - | | onTitleClick | Function | Trigger on clicking title. | - | | onOptionMenuClick | Function | Trigger on clicking extra icon of navigation bar. | 1.3.0 | | onPopMenuClick | Function | Trigger on clicking custom menu buttons in upper-right general menu. | 1.3.0 | | onPullDownRefresh | Function({from: `manual` | `code`}) | Trigger on pulling down page. | - | | onPullIntercept | Function | Trigger on pulling down interruption. | 1.11.0 | | onTabItemTap | Function | Trigger on clicking `tabItem`. | 1.11.0 | | onPageScroll | Function({scrollTop}) | Trigger on page scrolling. | - | | onReachBottom | Function | Trigger on pulling page till bottom. | - | | Others | Any | The developer can add any function or attribute column into the `object`. The `this` can be used for access in the page functions. | - |

Page Data Object

The initial `data` can be specified for the page by setting data. When data is an object, it is shared by all pages. In other words, when it returns and then enters the page again, the last page data will be displayed instead of the initial `data`. In such a case, the issue may be fixed by setting data as unchanged data or changing data as page exclusive `data`.

Set as unchanged data

copy

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

Set as page exclusive data (not recommended)

copy

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

Notes:

Do not modify `this.data` directly, which will not change the page status and will cause data inconsistency.

For example:

copy

```
Page({
 data: { arr:[] },
 doIt() {
   this.data.arr.push(1); // Do not do this!
   this.setData({arr: this.data.arr});
 }
});
```

Lifecycle Function

onLoad(query: Object)

Trigger on page initializing. It called only once for each page.

The query is the query object transferred in the `my.navigateTo` and `my.redirectTo`.

The query content is in the format: "parameter name=parameter value&parameter name=parameter value…"

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | query | Object | Parameter for opening the current page path. |

onShow()

Trigger on page showing or switching to foreground

onReady()

Trigger on completion of initial page rendering. It is called only once for each page, indicating the page is ready and can interact with view layer. For the setting of interface such as `my.setNavigationBar`, please set behind `onReady`.

onHide()

Trigger on page hiding or switching to background. Such as `my.navigateTo` to another page or switching via bottom tab.

onUnload()

Trigger on page unloading. Such as `my.redirectTo` or `my.navigateBack` to another page.

Page Event Handler

onShareAppMessage(options: Object)

Trigger on clicking the **Share** button in upper-right general menu or clicking in-page **Share** button.

Define the `onShareAppMessage` function in `Page` and set the sharing information:

- Display the **Share** button in the upper-right menu of every page by default. Only the shared content can be customized by using the `onShareAppMessage` function.
- The `onShareAppMessage` function is called when the user clicks the **Share** button.
- This event handler must return an Object to customize the shared content.
- The mini program supports to trigger the sharing by using the button component. The value of `open-type` is `share`.

Sample codes:

copy

```
// API-DEMO page/API/share/share.json
 {
        "defaultTitle" : "onShareAppMessage"
 }
```

copy

```
 <view class = "page" >
 <view class = "page-description" > Click the upper-right menu to
```

```
customize the sharing </view>
 </view>
```

copy

```
// API-DEMO page/API/share/share.js
 Page ({
  onShareAppMessage () {
  return {
    title : 'Sharing the View component' ,
    desc : 'The View component is general' ,
    path : 'page/component/view/view' ,
    };
  },
 });
```

Sample codes for triggering the sharing with the button component:

copy

```
<view>
<button type="primary" open-type="share" a:if="
{{canIUseShareButton}}">Share to friends</button>
</view>
```

copy

```
Page({
  data: { canIUseShareButton: true },
  setShareButtonSwitch () { this.setData({ canIUseShareButton:
my.canIUse('button.open-type.share') }) },
    onLoad() { this.setShareButtonSwitch(); } ,

    onShareAppMessage() {
    return {
      title: 'Mini program demo',
      desc: 'Mini program official demo that displays the supported
APIs and components',
      path: 'page/component/component-pages/view/view?param=123'
    }
  }
});
```

The parameters are in Object type and have the following attributes:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | from | String | Source of triggering sharing event. Valid values are:
- `button`: click the button in the page to trigger the sharing;
- `menu`: click the button in the upper-right menu to trigger the sharing;
- `code`: call the my.showSharePanel API to trigger the sharing. | | target | Object | If the value of `from` is `button`, `target` is the `button` that triggers the event. Otherwise, `button` is `undefined`. | | webViewUrl | String | When the page contains the web-view component, return the URL of the current web-view. |

This event handler must return an Object to customize the shared content.

Return value

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | title | String | Yes | Customized sharing title. Max 50 characters. | | desc | String | No | Customized description about the sharing. The maximum length is 140 characters when sharing to Sina Weibo, so it's suggested that the description does not exceed 140 characters. | | path | String | Yes | Customized sharing path. The customized parameters in the path can be obtained from the `onLoad` lifecycle function and follow the HTTP GET rules. The path cannot contain the root directory (/). | | imageUrl | String | No | The path of the customized icon, which can be a web image path.
Recommended Image size is 1200 x 630 pixels and should not be more than 8M, and Minimum image size is 200 x 200 pixels. | | bgImgUrl | String | No | The path of the customized image, which can be a web image path. The image size is suggested to be 750 x 825 pixels. | | success | Function | No | The callback method that indicates a successful sharing. | | fail | Function | No | The callback method that indicates a failed sharing. |

Success callback function

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | channelName | String | The sharing channel. | | shareResult | Boolean | The result that indicates whether the sharing is successful. |

onTitleClick()

Trigger on clicking title.

onOptionMenuClick()

Trigger on clicking upper-right corner menu button.

onPopMenuClick()

Trigger on clicking upper-right corner general menu button.

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

Trigger on pulling down to refresh. It is required to enable `pullRefresh` in the `window` option of app.json. When the data refresh is processed completely, call `my.stopPullDownRefresh` to stop the pull-to-refresh for that page.

onPullIntercept()

Trigger on pulling down interruption.

onTabItemTap(object: Object)

Trigger on clicking `tabItem`

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | from | String | Click source. | | pagePath | String | Page path of the clicked tabItem. | | text | String | Button text of the clicked tabItem. | | index | Number | Number of the clicked tabItem, starting from 0. |

onPageScroll({scrollTop})

Trigger on page scrolling, `scrollTop` is the page scrolling distance.

onReachBottom()

Trigger on pulling page till bottom.

Events

To simplify codes, a new `event` handler object events is available. The existing page handler is equivalent to the exposed event functions on the page instance.

Notes:

- The support for events starts from basic library version 1.13.7.
- Please distinguish the basic library version requirements for the same named functions of the page event handler and events.

Below is the list of event functions supported by events:

| | | | | | --- | --- | --- | --- | | **Event** | **Type** | **Description** | **Lowest version** | | onBack | Function | Trigger on page returning. | 1.13.7 | | onKeyboardHeight | Function | Trigger on keyboard height changing. | 1.13.7 | | onOptionMenuClick | Function | Trigger on clicking upper-right corner menu button. | 1.13.7 | | onPopMenuClick | Function | Trigger on clicking upper-right corner general menu button. | 1.13.7 | | onPullIntercept | Function | Trigger on pulling down interruption. | 1.13.7 | | onPullDownRefresh | Function({from: manual/code}) | Trigger on pulling down page. | 1.13.7 | | onTitleClick | Function | Trigger on clicking title. | 1.13.7 | | onTabItemTap | Function | Trigger on click non-current tabItem. | 1.13.7 | | beforeTabItemTap | Function | Trigger before click non-current tabItem. | 1.13.7 | | onResize | Function({size: {windowWidth: number, windowHeight: number}}) | Trigger on window size changing. | 1.16.0 |

Sample code:

copy

```
// Feature detection
my.canIUse('page.events.onBack');

Page({
  data: {
    text: 'This is page data.'
  },
  onLoad(){
    // trigger on page loading
  },
```

```
  events:{
    onBack(){
      // Trigger on page returning
    },
    onKeyboardHeight(e){
      // Trigger on keyboard height changing
      console.log('keyboard height:', e.height)
    },
    onOptionMenuClick(){
      // Trigger on clicking upper-right corner menu button
    },
    onPopMenuClick(e){
      // Trigger on clicking custom menu buttons in upper-right
general menu
      console.log('index of the clicked custom menu', e.index)
      console.log('name of the clicked custom menu', e.name)
      console.log('menuIconUrl of the clicked custom menu',
e.menuIconUrl)
    },
    onPullIntercept(){
      // Trigger on pulling down interruption
    },
    onPullDownRefresh(e){
      // Trigger on pulling down page The e.from value “code”
indicates the event triggered by startPullDownRefresh; value “manual”
indicates the pull-down event trigger by user
      console.log('type of triggered pull-down refresh', e.from)
      my.stopPullDownRefresh()
    },
    onTitleClick(){
      // Trigger on clicking title
    },
    onTabItemTap(e){
      // e.from means triggering after clicking tabItem and switching;
value “user” indicates event triggered by user clicking; value “api”
indicates event triggered by switchTab
      console.log('type of triggering tab change', e.from)
      console.log('path of page corresponding to the clicked tab',
e.pagePath)
      console.log('text of the clicked tab', e.text)
      console.log('index of the clicked tab', e.index)
    },
    beforeTabItemTap(){
      // trigger on clicking tabItem but before switching
    },
    onResize(e){
      // Trigger on window size changing
      var {windowWidth, windowHeight} = e.size
      console.log('width of changed window', windowWidth)
      console.log('height of changed window', windowHeight)
    },
```

```
  }
})
```

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

The `setData` sends data from logic layer to view layer and changes the value of `this.data`.

The `Object` is expressed in the form `key: Value`.. The `key` value in `this.data` is changed to `value`. Here, the `key` can be flexibly provided in form of data path, such as `array[2].message`, `a.b.c.d`. It is not necessary to predefine in `this.data`.

The following points are worth attentions in use:

1. It is invalid to modify `this.data` directly, which will not change the page status and will cause data inconsistency.
2. Only the JSON supported data is supported.
3. Try not to set too many data once.
4. Do not set any value in the data as undefined, otherwise, that item will not be set, and potential issue may arise.

Sample code:

copy

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

copy

```
Page({
  data: {
    text: 'test',
    array: [{text: 'a'}],
    object: {
      text: 'blue',
    },
    name: 'Mini Program',
  },
  changeTitle() {
```

```
    // Wrong! Do not modify the data directly
    // this.data.text = 'changed data'

    // Correct!
    this.setData({
      text: 'ha',
    });
  },
  changeArray() {
    // Possible to modify data by using directly data path
    this.setData({
      'array[0].text': 'b',
    });
  },
  changePlanetColor(){
    this.setData({
      'object.text': 'red',
    });
  },
  addNewKey() {
    this.setData({
      'newField.text': 'c',
    });
  },
  changeName() {
    this.setData({
      name: 'Mini Program',
    }, () => { // Accept transfer of callback function
      console.log(this); // this: current page instance
      this.setData({ name: this.data.name + ', ' + 'welcome!'});
    });
  },
});
```

Parameter description:

| | | | | | --- | --- | --- | --- | | **Event** | **Type** | **Description** | **Lowest version** | | data | Object | Data to be changed. | - | | callback | Function | Callback function, to be executed on completion of page rendering and update. | 1.7.0, Use `my.canIUse('page.setData.callback')` for compatibility processing. |

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

Note: `$spliceData` is supported since version 1.7.2. The **my.canIUse('page.$spliceData')** can be used for compatibility processing.

Similarly, the `spliceData` is used to transfer data from logic layer to view layer, but has higher performance than `setData` in processing long list.

The `Object` is expressed in the form `key: Value`.. The `key` value in `this.data` is changed to `value`. Here, the key can be flexibly provided in form of data path, such as `array[2].message`, `a.b.c.d`. It is not necessary to predefine in `this.data`. The `value` is an array (format: [start, deleteCount, ...items]). The first element of the array is the start position of the operation, the second element is the number of elements to be deleted, and other other elements are the insertion data. It maps the array `splice` method in `es5`.

Sample code:

copy

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

copy

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

Page output:

copy

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

Parameter description:

| | | | --- | --- | --- | | **Event** | **Type** | **Description** | | data | Object | Data to be changed. | | callback | Function | Callback function, to be executed on completion of page rendering and update. |

Page.prototype.$batchedUpdates(callback: Function)

Batch update data.

Note: `$batchedUpdates` is supported since version 1.14.0. Themy.canIUse('page.$batchedUpdates') can be used for compatibility processing.

Parameter description:

| | | | | --- | --- | --- | | **Event** | **Type** | **Description** | | callback | Function | The data operation in the callback function will be updated in batch. |

Sample code:

copy

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

copy

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

1. In this example, page `counter` adds 2 on each button clicking.
2. The `setData` is placed within this.$batchedUpdates. Thus, only one data transfer happens despite of multiple `setData`.

Page.route

Path of `Page`, mapping the path value configured in app.json, type `String`

> This is a read-only attribute.

copy

```
Page({
  onShow() {
    // Map the path value configured in app.json
    console.log(this.route)
  }
})
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_page_page-mechanism

---

Page Structure {#page-structure}

Last updated: 2022-07-03

Path: miniprogram_gcash

Page Structure

2022-07-03 18:44

The page structure is defined by he `.axml` file in the `pages` directory, the content should follow the `axml` syntax.

AXML is similar to HTML, and there are also some differences, please refer to AXML for more detail information.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_page_page-structure

---

Page Structure {#page-structure}

Last updated: 2021-05-10

Path: miniprogram_gcash

Page Structure

2021-05-10 03:43

The page structure is defined by he `.axml` file in the `pages` directory, the content should follow the `axml` syntax.

AXML is similar to HTML, and there are also some differences, please refer to AXML for more detail information.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_page_page-structure

---

Page Style {#page-style}

Last updated: 2022-07-03

Path: miniprogram_gcash

Page Style

2022-07-03 18:44

The page style is defined by the `.acss` file in the `/pages` directory.

Each page will have a root element `page`, the background color or page height can be configured, for example.

copy

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

The detail information about acss can be referred in ACSS.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_page_page-style

---

Page Style {#page-style}

Last updated: 2021-05-09

Path: miniprogram_gcash

Page Style

2021-05-09 18:43

The page style is defined by the `.acss` file in the `/pages` directory.

Each page will have a root element `page`, the background color or page height can be configured, for example.

copy

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

The detail information about acss can be referred in ACSS.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_page_page-style

---

PageResult {#pageresult}

Last updated: 2022-07-03

Path: miniprogram_gcash

PageResult

2022-07-03 18:44

Fault page.

Sample Code

copy

```
// API-DEMO page/component/page-result/page-result.json
{
  "defaultTitle": "fault feedback",
  "usingComponents": {
    "page-result": "mini-antui/es/page-result/index"
  }
}
```

copy

```
<!-- API-DEMO page/component/page-result/page-result.axml -->
<page-result
  type="network"
  title="Network is poor"
```

```
    brief="It looks like the furthest distance in the world"
  />
  <page-result
    type="network"
    title="Network is poor"
    brief="It looks like the furthest distance in the world"
  >
    <view class="am-page-result-btns">
      <view onTap="backHome">Back home</view>
      <view>Sample button</view>
    </view>
  </page-result>
```

copy

```
// API-DEMO page/component/page-result/page-result.js
Page({
  backHome() {
    my.navigateBack();
  }
});
```

copy

```
.am-page-result {
  display: flex;
  flex-direction: column;
}
.am-page-result-btns {
  flex: 1;
  display: flex;
  flex-direction: column;
  justify-content: flex-end;
  align-content: center;
  padding-bottom: 100rpx;
}
.am-page-result-btns > view {
  color: #108EE9;
  font-size: 40rpx;
  margin-top: 52rpx;
  text-align: center;
}
```

Attributes

| | | | | | | --- | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Default** | **Required** | | type | Fault page type: network fault - network, service busy - busy, service abnormity - error, empty status - empty, user logoff - logoff. | String | network | No | | local | Is local fault content or not. | Boolean | false | No | | title | Fault prompt title. | String | - | No | | brief | Fault prompt brief. | String | - | No |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_result_pageresult

---

Pagination {#pagination}

Last updated: 2022-07-03

Path: miniprogram_gcash

Pagination

2022-07-03 18:44

Pagination

Sample Code

copy

```
{
  "defaultTitle": "Mini Program AntUI component library",
  "usingComponents": {
    "pagination": "mini-antui/es/pagination/index"
  }
}
```

copy

```
<view>
  <view class="demo-title">Basic usage</view>
  <pagination total="{{20}}" current="{{1}}"/>
  <view class="demo-title">Arrow button</view>
  <pagination mode="icon" total="{{20}}" current="{{10}}"/>
  <view class="demo-title">Simple mode</view>
  <pagination simple total="{{20}}" current="{{1}}"/>
  <view class="demo-title">Button disabled</view>
  <pagination total="{{20}}" current="{{1}}" disabled/>
  <view class="demo-title">Custom button text</view>
  <pagination arrow prevText="Previous" nextText="Next" total="
{{20}}" current="{{1}}"/>
</view>
```

copy

```
Page({})
```

Attributes

| | | | | | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Default** | | mode | Button form options: text, icon. | String | text | | total | Total number of pages. | Number | 0 | | current | Current page number. | Number | 0 | | simple | Hide value or not. | Boolean | false | | disabled | Disabled status. | Boolean | false | | prevText | Text for page-up button. | String | Previous page | | nextText | Text for page-down button. | String | Next page | | btnClass | Pagination button style, for text type buttons only. | String | - | | onChange | Pagination callback function. | (index: Number) => void | - |

Note:

prevText and nextText take effect only when mode is text.

九色鹿

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_layout-navigation_pagination

---

Payment capability {#payment-capability}

Last updated: 2022-07-03

Path: miniprogram_gcash

Payment capability

2022-07-03 18:44

Users can trigger the wallet cashier page on a mini program. The payment process and user experience on the mini program are similar to those of the native app.

Prerequisites

This capability is open to a merchant with a valid business license that is verified by its wallet. The merchant website should be accessible and provide clear business content and complete product information.

User experience

The overall payment process includes the following steps:

1. A user selects a product in the mini program and places an order.

2. The user confirms the purchase and enters the payment page that is triggered by the mini program.
3. The user confirms the payee and amount on the checkout page, then confirms the payment.
4. The payment success page is displayed.

Procedures

To develop the payment capability, follow the steps below:

1. Create a mini program

The merchant/ISV gets started with the workspace and publishes a mini program in the Mini Program Platform. For more information, see the product guide overview and the developer guide.

2. Add features (Optional)

By default, the payment capability is available. For other features, you need to add a feature and define the details according to your business requirements. For more information, see Features.

3. Call APIs

1. The wallet user creates a payment order in a mini program.
2. The merchant or ISV server creates the order by calling the `/{version}/payments/pay` OpenAPI from the wallet server.
3. The wallet server returns parameters such as `acquirementId` and checkout URL to the mini program.
4. The mini program calls the `tradePay` JSAPI by triggering the wallet payment process and other return parameters and then gets the response.
5. The user confirms the payment. Then the wallet server calls the `/{version}/payments/notifyPayment` OpenAPI and sends the order status notification to the min program server.
6. The user is redirected to the payment result page in the mini program.

Note:

- The `version` is the version of Open APIs, for example, `v1` or `v2`.
- The parameter `userId` or `uid` is fetched by calling the `applyToken` OpenAPI. For more information, see User information capability.

API list

| | | | --- | --- | | **JSAPI** | **Description** | | my.tradePay | Trigger the cashier page from the wallet. | | **OpenAPI** | **Description** | | / `{version}/payments/{apiName}` The `version` is the version of Open APIs, for example, `v1` or `v2`. | For details, see the Open APIs for Merchants chapter. |

More information

Capabilities

JSAPIs

Open APIs

Developing Mini Program

Using Mini Program Platform

Features

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/capability-payment

---

Performance {#performance}

Last updated: 2022-07-08

Path: miniprogram_gcash

Performance

2022-07-08 02:08

You can see the performance with historical data about the mini program. You can select one day before the current date, and see the following numbers on different card pages:

- **Unique Visitor by User**

The number of unique visitors on the selected date that are calculated by the user ID.

- **Unique Visitor by Device**

The number of unique visitors on the selected date that are calculated by the device ID.

- **Page View**

The number of page views on the selected date.

- **New Users**

The number of new users on the selected date.

- **7-Day Active Users**

The number of active users within the past 7 days till the selected date.

- **Accumulated Users**

The number of users that are accumulated from the first day of the mini program release to the selected date.

For each card, you can also see the growth rates, including the daily growth rate, week-over-week growth rate, and month-over-month growth rate.

Next steps

Real-Time Analysis

More information

Analytics

Overview

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/platform/analytics_performance

---

PickerItem {#pickeritem}

Last updated: 2022-07-03

Path: miniprogram_gcash

PickerItem

2022-07-03 18:44

Selection input.

Sample Code

copy

```
// API-DEMO page/component/input-item/input-item.json
{
```

```
  "defaultTitle": "Mini Program AntUI component library",
  "usingComponents": {
    "list": "mini-antui/es/list/index",
    "list-item": "mini-antui/es/list/list-item/index",
    "input-item": "mini-antui/es/input-item/index",
    "picker-item": "mini-antui/es/picker-item/index"
  }
}
```

copy

```
<!-- API-DEMO page/component/input-item/input-item.axml -->
<view>
  <view style="margin-top: 10px;" />
  <list>
    <input-item
      data-field="cardNo"
      clear="{{true}}"
      value="{{cardNo}}"
      className="dadada"
      placeholder="Bank card number"
      focus="{{inputFocus}}"
      onInput="onItemInput"
      onFocus="onItemFocus"
      onBlur="onItemBlur"
      onConfirm="onItemConfirm"
      onClear="onClear"
    >
      Card number
      <view slot="extra" class="extra" onTap="onExtraTap"></view>
    </input-item>
    <picker-item
      data-field="bank"
      placeholder="Select issuing bank"
      value="{{bank}}"
      onPickerTap="onPickerTap"
    >
      Issuing bank
    </picker-item>
    <input-item
      data-field="name"
      placeholder="Name"
      type="text"
      value="{{name}}"
      clear="{{true}}"
      onInput="onItemInput"
      onClear="onClear"
    >
      Name
    </input-item>
    <input-item
      data-field="password"
```

```
        placeholder="Password"
        password
      >
        Password
      </input-item>
      <input-item
        data-field="remark"
        placeholder="Remarks"
        last="{{true}}"
      />
    </list>
    <view style="margin: 10px;">
      <button type="primary" onTap="onAutoFocus">Focus</button>
    </view>
  </view>
```

copy

```
// API-DEMO page/component/input-item/input-item.js
const banks = ['Mybank', 'CCB', 'ICBC', 'SPDB']

Page({
  data: {
    cardNo: '1234****',
    inputFocus: true,
    bank: '',
    name: '',
  },
  onAutoFocus() {
    this.setData({
      inputFocus: true,
    });
  },
  onExtraTap() {
    my.alert({
      content: 'extra tapped',
    });
  },
  onItemInput(e) {
    this.setData({
      [e.target.dataset.field]: e.detail.value,
    });
  },
  onItemFocus() {
    this.setData({
      inputFocus: false,
    });
  },
  onItemBlur() {},
  onItemConfirm() {},
  onClear(e) {
    this.setData({
```

```
          [e.target.dataset.field]: '',
        });
      },
      onPickerTap() {
        my.showActionSheet({
          title: 'Select issuing bank',
          items: banks,
          success: (res) => {
            this.setData({
              bank: banks[res.index],
            });
          },
        });
      },
    });
```

copy

```
    /* API-DEMO page/component/input-item/input-item.acss */
    .extra {
      background-image: url('https://img.example.com/example.svg');
      background-size: contain;
      background-repeat: no-repeat;
      background-position: right center;
      opacity: 0.2;
      height: 20px;
      width: 20px;
      padding-left: 10px;
    }
```

Attributes

| | | | | | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Default** | | className | Customized class. | String | - | | labelCls | Customized label class. | String | - | | pickerCls | Customized selection region class. | String | - | | last | Is the last row or not. | Boolean | false | | value | Initial contents. | String | - | | name | Component name, used for getting data via form submission. | String | - | | placeholder | Placeholder. | String | - | | onPickerTap | Trigger on clicking pickeritem. | (e: Object) => void | - |

Slots

| | | | | --- | --- | --- | | **slotname** | **Description** | **Required** | | extra | Used to render the description right to picker-item. | No |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_form_pickeritem

---

Platform User's Guide {#platform-user's-guide}

Path: miniprogram_gcash

Platform User's Guide

This product guide is intended for mini program platform users to learn the product features and procedures to manage the whole lifecycle of mini programs.

Mini Program is a new technology that helps you quickly develop high-quality services and grow your business on mobile apps with better user experience. With the Mini Program Development Platform, you can manage the whole lifecycle of mini programs. You can either directly create, develop, upload, release or remove mini programs, or authorize developers to manage min programs with an approval process. You can also see the data with analytics and quality functionality to improve efficiency.

Mini Program Development Platform

See the details of all the available features to manage the whole lifecycle of a mini program. Learn more

- How to manage mini programs
- How to transform an HTML 5 mobile app to an HTML 5 mini program

Mini Program Operation Platform

See the features related to marketing and operations to manage the business operations of a mini program.

- How to manage notification templates (Super apps)\\\ Please log in to continue
- How to navigate to Operation Platform (Operators)\\\ Please log in to continue

Related Topics

Learn the basic steps for onboarding before you start building your own Mini Programs.

Get the available reference resources, s

Dip into the rich UI guidelines to design a Mini Program.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/platform/overview

---

Popover {#popover}

Last updated: 2022-07-03

Path: miniprogram_gcash

Popover

2022-07-03 18:44

Bubble It is possible to set Popover-item width and height to change the bubble size. Text adaptive width & height is not supported.

Note: The setting popover is located right below the specific element. It is possible to place the element within the popover and set the position as bottom.

Sample Code

copy

```
// API-DEMO page/component/popover.json
{
  "defaultTitle": "Popover",
  "usingComponents": {
    "popover": "mini-antui/es/popover/index",
    "popover-item": "mini-antui/es/popover/popover-item/index"
  }
}
```

copy

```
<!-- API-DEMO page/component/popover/.axml-->
<view class="demo-popover">
  <popover
    position="{{position}}"
    show="{{show}}"
    showMask="{{showMask}}"
    onMaskClick="onMaskClick"
  >
    <view class="demo-popover-btn" onTap="onShowPopoverTap">Click
{{show ? 'hide' : 'show'}}</view>
    <view slot="items">
      <popover-item onItemClick="itemTap1">
        <text>{{position}}</text>
      </popover-item>
      <popover-item onItemClick="itemTap2">
        <text>line2</text>
      </popover-item>
    </view>
  </popover>
</view>
<view class="demo-popover-test-btns">
  <button class="demo-popover-test-btn"
onTap="onNextPositionTap">Next position</button>
  <button class="demo-popover-test-btn"
```

```
onTap="onMaskChangeTap">Mask{{showMask ? 'hide' : 'show'}}</button>
    </view>
```

copy

```
    // API-DEMO page/component/popover.js
    const position = ['top', 'topRight', 'rightTop', 'right',
'rightBottom', 'bottomRight', 'bottom', 'bottomLeft', 'leftBottom',
'left', 'leftTop', 'topLeft'];
    Page({
      data: {
        position: position[0],
        show: false,
        showMask: true,
      },
      onShowPopoverTap() {
        this.setData({
          show: !this.data.show,
        });
      },
      onNextPositionTap() {
        let index = position.indexOf(this.data.position);
        index = index >= position.length - 1 ? 0 : index + 1;
        this.setData({
          show: true,
          position: position[index],
        });
      },
      onMaskChangeTap() {
        this.setData({
          showMask: !this.data.showMask,
        });
      },
      onMaskClick() {
        this.setData({
          show: false,
        });
      },
      itemTap1() {
        my.alert({
          content: 'Click1',
        });
      },
      itemTap2() {
        my.alert({
          content: 'Click2',
        });
      },
    });
```

copy

```
/* API-DEMO page/component/popover.css */
.demo-popover {
  display: flex;
  align-items: center;
  justify-content: center;
  width: 100%;
  height: 400px;
}
.demo-popover-btn {
  width: 100px;
  height: 100px;
  line-height: 100px;
  text-align: center;
  background-color: #fff;
  border: 1px solid #dddddd;
  border-radius: 2px;
}
.demo-popover-test-btns {
  display: flex;
  justify-content: space-around;
}
.demo-popover-test-btn {
  width: 45%;
}
```

Attributes

| | | | | | | --- | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Default** | **Required** | | className | Outmost layout style. | String | - | No | | show | Show bubble or not. | Boolean | false | Yes | | showMask | Show mask or not. | Boolean | true | No | | position | Bubble position options: top, topRight, topLeft, bottom, bottomLeft, bottomRight, right, rightTop, rightBottom, left, leftBottom, leftTop. | String | bottomRight | No |

popover-item

| | | | | --- | --- | --- | | **Property** | **Description** | **Type** | | className | Single item style. | String | | onItemClick | Single item click event. | () => void |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_floating-layer_popover

---

Popup {#popup}

Last updated: 2022-07-03

Path: miniprogram_gcash

Popup

2022-07-03 18:44

Popup window.

| | | | | | | --- | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Default** | **Required** | | className | Custom class. | String | | No | | show | Show menu or not. | Boolean | false | No | | animation | Enable animation or not. | Boolean | true | No | | mask | Show mask or not. Clicking outside does not trigger onClose when it is not shown. | Boolean | true | Yes | | position | Control the direction in which the menu pops up. Bottom indicating the bottom side, left the left side, top the top side and right the right side. | String | bottom | No | | disableScroll | Disable page scroll or not when mask is shown. | Boolean | true | No | | zIndex | Define the number of popup levels. | Number | 0 | No |

Slots

It is possible to define the parts to be shown in the popup component. See the following example for details.

Example

copy

```
{
  "defaultTitle": "AntUI Component Library",
  "usingComponents": {
    "popup": "mini-antui/es/popup/index"
  }
}
```

copy

```
<view>
  <view class="btn-container">
    <button onTap="onTopBtnTap">Popup</button>
  </view>
  <popup show="{{showTop}}" position="top" onClose="onPopupClose">
    <view style="height: 200px; background: #fff; display: flex;
justify-content: center; align-items: center;">hello world</view>
  </popup>
</view>
```

copy

```
Page({
  data: {
    showTop: false,
  },
```

```
  onTopBtnTap() {
    this.setData({
      showTop: true,
    });
  },
  onPopupClose() {
    this.setData({
      showTop: false,
    });
  },
});
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_floating-layer_popup

---

Publish an event {#publish-an-event}

Last updated: 2022-07-07

Path: miniprogram_gcash

Publish an event

2022-07-07 17:08

After you complete defining an event, you can choose **Save & Publish** to publish the event. The event configuration is then completed and the system will start to retrieve data in about 5 minutes.

Note: After you publish an event, the **Event Name** and **Data Reporting Method** fields cannot be changed but other fields can still be modified.

Next steps

Analyze events and funnels

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/platform/publish-event

---

Quality {#quality}

Last updated: 2022-07-07

Path: miniprogram_gcash

Quality

2022-07-07 17:08

Quality is a real-time crash reporter that helps you track, prioritize, and fix stability issues that erode your mini program quality.

Features

You can perform the following activities within one or all version of a mini program:

- See the following statistics in a form of a graph within different time ranges:
- `Total HTTP Requests`: The total number of HTTP requests.
- `JSAPI Call Volume`: The total number of JSAPI calls.
- `JSError Quantity`: The total number of JS errors.
- `Abnormal Requests`: The total number of abnormal requests of HTTP requests or JSAPI calls.
- `Affected Users`: The total number of users that are affected by JSErrors or loading exceptions.
- See the details of abnormal requests, and errors:
- `Abnormal Request Details`: You can see abnormal URLs, error descriptions, error codes, and a total number of errors for each abnormal URL.
- `JSError Details`: You can see the error stack description, total number of events caused by each JSError, and total number of affected users caused by each JS error.
- `Abnormal Resource Files Details`: You can see abnormal URLs, error types, and a total number of errors for each abnormal URL.

More information

Overview

Workflow Procedures

Member Role

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/platform/analytics-quality

---

Quick start {#quick-start}

Last updated: 2021-05-09

Path: miniprogram_gcash

Quick start

2021-05-09 18:43

This tutorial is designed to quickly get you started with developing a Mini Program quickly.

You can see the whole lifecycle of a mini program in the following figure:

Figure 1. Mini program lifecycle

Mini program development overview

The life-cycle of a mini program covers from the developer account creation to the release of the mini program.

Prerequisite

Before you can get started, make sure you have completed the following settings:

1. Apply for an account to join the Mini Program Platform

As a developer, you can join the platform to be a Mini Program admin or a Mini Program developer after you are receive an email invitation.

2. Complete the on-boarding process
3. Create a mini program

- If you are a Mini Program admin, you can apply to create Mini Program on the Mini Program Platform and submit the Mini Program particulars, including but not limited to: name, description, logo image, etc. After successful creation, a unique identifier will be assigned to the newly created Mini Program.
- If you are a Mini Program developer, the Mini Program admin can add you as a member of a Mini Program.

4. Dowload Mini Program Studio

Develop & Debug in IDE

1. Create a project in Mini Program Studio
2. Link the project with the mini program that is created in the Mini Program Platform.
3. Code and debug
4. Upload the code package from Mini Program Studio to the Mini Program Platform.

Review & Release

1. Review the mini program and request to release the mini progrm
2. Workspace admin reviews and approves the release request.
3. Release the mini program.

Users can now open the mini program in the app.

Also, marketing capabilities can be enabled to the mini program for operational scenarios. And the life-cycle of a mini program ends when a mini program is removed from the Mini Program Platform.

Next steps

Developers can learn more about Mini Program in the Quick Start documentation:

- Apply for an account.
- Run the first Mini Program quickly by using the demo available in the Mini Program studio.
- Know more about the Mini Program project structure.
- According to the project structure knowledge, learn more about the demo Mini Program.
- Learn how to publish a Mini Program.

九色鹿

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/quick-start_overview

---

Quick start {#quick-start}

Last updated: 2022-07-03

Path: miniprogram_gcash

Quick start

2022-07-03 18:44

This tutorial is designed to quickly get you started with developing a Mini Program quickly.

You can see the whole lifecycle of a mini program in the following figure:

Figure 1. Mini program lifecycle

Mini program development overview

The life-cycle of a mini program covers from the developer account creation to the release of the mini program.

Prerequisite

Before you can get started, make sure you have completed the following settings:

1. Apply for an account to join the Mini Program Platform

As a developer, you can join the platform to be a Mini Program admin or a Mini Program developer after you are receive an email invitation.

2. Complete the on-boarding process
3. Create a mini program

- If you are a Mini Program admin, you can apply to create Mini Program on the Mini Program Platform and submit the Mini Program particulars, including but not limited to: name, description, logo image, etc. After successful creation, a unique identifier will be assigned to the newly created Mini Program.
- If you are a Mini Program developer, the Mini Program admin can add you as a member of a Mini Program.

4. Dowload Mini Program Studio

Develop & Debug in IDE

1. Create a project in Mini Program Studio
2. Link the project with the mini program that is created in the Mini Program Platform.
3. Code and debug
4. Upload the code package from Mini Program Studio to the Mini Program Platform.

Review & Release

1. Review the mini program and request to release the mini progrm
2. Workspace admin reviews and approves the release request.
3. Release the mini program.

Users can now open the mini program in the app.

Also, marketing capabilities can be enabled to the mini program for operational scenarios. And the life-cycle of a mini program ends when a mini program is removed from the Mini Program Platform.

Next steps

Developers can learn more about Mini Program in the Quick Start documentation:

- Apply for an account.
- Run the first Mini Program quickly by using the demo available in the Mini Program studio.
- Know more about the Mini Program project structure.
- According to the project structure knowledge, learn more about the demo Mini Program.
- Learn how to publish a Mini Program.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/quick-start_overview

---

Real-time analysis {#real-time-analysis}

Last updated: 2022-07-07

Path: miniprogram_gcash

Real-time analysis

2022-07-07 17:08

You can see the real-time data about the mini program on the current date. You can also select one day before the current date or a time period from any date in the past week to the current date.

Features

You can benefit from the following features:

- **Overview**

You can see a data overview of the current date and daily growth rates of the following items:

- Number of users calculated by the user ID
- Number of users calculated by the device ID
- Page visits

It's also supported to select one day before the current date to view the data.

- **Page Visits**

You can select a time range to check the performance of every single page in a table, which includes the following fields:

- Page Path
- Visits
- Visitors
- New Visits by Shared Users
- Time on Page (sec)
- Shared Users
- Total Shares

- **Version Adoption**

You can select a time range to see the ratio of different mini program versions that are used by users.

- **Success Rate of JSAPI Calls**

You can see the payment success rate that is calculated by the total number of payment requests and the number of successful payments. The following JSAPIs are called for this capability:

- my.tradepay
- getAuthCode
- getOpenUserInfo

- **JSAPI errors**

You can see the error details, such as error code, description, number and time of occurrence, and so on.

The following JSAPIs are called for this capability:

- my.tradepay
- getAuthCode
- getOpenUserInfo

Next steps

Manage Events

More information

Analytics

Performance

Overview

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/platform/analytics_realtime

---

Register Mini Program {#register-mini-program}

Last updated: 2022-07-03

Path: miniprogram_gcash

Register Mini Program

2022-07-03 18:44

App(Object)

`App()` is used to register the Mini Program, accepts an object as the parameter to configure the lifecycle of Mini Program. `App()` should be called in app.js and only be called once.

Object Parameter Description

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Description** | **Trigger** | | onLaunch | Function | Listening to Mini Program initialization. | On completion of Mini Program initialization, invoked only once. | | onShow | Function | Listening to Mini Program showing. | On startup of Mini Program or switching to foreground from background. | | onHide | Function | Listening to Mini Program hiding. | On switching Mini Program from foreground to background. | | onError | Function | Listening to Mini Program error. | On js error of the Mini Program. | | onUnhandledRejection | Function | Listen for the *unhandledrejection* event. | Triggered when a JavaScript `Promise` that has no rejection handler is rejected. |

Foreground/background definition:

- When the user leaves mobile app with the close button at upper-right corner or the device Home button, the Mini Program is not directly destroyed but switched to the background.
- When mobile app is started or the Mini Program is opened again, it is switched to the foreground from the background.
- Only when the Mini Program stays in background for a certain time or occupies too many system resources, it is destroyed.

onLaunch/onShow Options Parameter Description

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | query | Object | Current Mini Program query, parsed from the query field in the startup parameter. | | path | String | Current Mini Program page address, parsed from the page field in the startup parameter, home page by default when page is ignored. | | referrerInfo | Object | Source information. |

- This parameter can be obtained from the `onLaunch` method upon the first-time Mini Program startup
- The parameter can also be obtained from the `onShow` method when the Mini Program in background is reopened with schema.

copy

```
App({
  onLaunch(options) {
    // first opening
    console.log(options.query);
    // {number:1}
  },
  onShow(options) {
    // reopening with schema from background
    console.log(options.query);
    // {number:1}
  },
})
```

referrerInfo attribute description

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Description** | **Compatibility** | | appId | string | Source Mini Program. | | | sourceServiceId | String | Source plug-in, visible in the plug-in running mode. | 1.11.0 | | extraData | Object | Data transferred from the source Mini Program. | |

Notes:

- Do not operate page stack like redirectTo/navigateTo on the onShow.
- The basic library version used in AppContainer currently is 1.14.2.

onHide()

The `onHide()` method will be triggered when Mini Program changes to background from foreground.

Sample code

copy

```
App({
  onHide() {
    // when changes to background
    console.log('app hide');
  },
});
```

onError()

The `onError()` method will be triggered when script error happens.

Sample code

copy

```
App({
  onError(error) {
    // the Mini Program script error happens
    console.log(error);
  },
});
```

onUnhandledRejection()

The onUnhandledRejection() method will be triggered when a JavaScript `Promise` that has no rejection handler is rejected.

Sample code

copy

```
App({
  onUnhandledRejection(res) {
    // A JavaScript Promise that has no rejection handler is rejected.
    console.log(res.reason, res.promise);
    //res.reason describes the rejection reason and res.promise
describes the rejected Promise.
  },
});
```

Global Data

Global data can be configured in `App()`. Other pages can get and modify the global data directly.

Sample code

copy

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

FAQ

Q: Can Mini Program be closed in app.js?

A: No, Mini Program can only be closed by clicking close button in the top right corner.

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_app_register-mini-program

---

Register Mini Program {#register-mini-program}

Last updated: 2021-05-09

Path: miniprogram_gcash

Register Mini Program

2021-05-09 18:43

App(Object)

`App()` is used to register the Mini Program, accepts an object as the parameter to configure the lifecycle of Mini Program. `App()` should be called in app.js and only be called once.

Object Parameter Description

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Description** | **Trigger** | | onLaunch | Function | Listening to Mini Program initialization. | On completion of Mini Program initialization, invoked only once. | | onShow | Function | Listening to Mini Program showing. | On startup of Mini Program or switching to foreground from background. | | onHide | Function | Listening to Mini Program hiding. | On switching Mini Program from foreground to background. | | onError | Function | Listening to Mini Program error. | On js error of the Mini Program. | | onUnhandledRejection | Function | Listen for the *unhandledrejection* event. | Triggered when a JavaScript `Promise` that has no rejection handler is rejected. |

Foreground/background definition:

- When the user leaves mobile app with the close button at upper-right corner or the device Home button, the Mini Program is not directly destroyed but switched to the background.
- When mobile app is started or the Mini Program is opened again, it is switched to the foreground from the background.
- Only when the Mini Program stays in background for a certain time or occupies too many system resources, it is destroyed.

onLaunch/onShow Options Parameter Description

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | query | Object | Current Mini Program query, parsed from the query field in the startup parameter. | | path | String | Current Mini Program page address, parsed from the page field in the startup parameter, home page by default when page is ignored. | | referrerInfo | Object | Source information. |

- This parameter can be obtained from the `onLaunch` method upon the first-time Mini Program startup
- The parameter can also be obtained from the `onShow` method when the Mini Program in background is reopened with schema.

copy

```
App({
  onLaunch(options) {
    // first opening
    console.log(options.query);
    // {number:1}
  },
  onShow(options) {
    // reopening with schema from background
    console.log(options.query);
    // {number:1}
  },
})
```

referrerInfo attribute description

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Description** | **Compatibility** | | appId | string | Source Mini Program. | | | sourceServiceId | String | Source plug-in, visible in the plug-in running mode. | 1.11.0 | | extraData | Object | Data transferred from the source Mini Program. | |

Notes:

- Do not operate page stack like redirectTo/navigateTo on the onShow.
- The basic library version used in AppContainer currently is 1.14.2.

onHide()

The `onHide()` method will be triggered when Mini Program changes to background from foreground.

Sample code

copy

```
App({
  onHide() {
    // when changes to background
    console.log('app hide');
```

```
  },
});
```

onError()

The `onError()` method will be triggered when script error happens.

Sample code

copy

```
App({
  onError(error) {
    // the Mini Program script error happens
    console.log(error);
  },
});
```

onUnhandledRejection()

The onUnhandledRejection() method will be triggered when a JavaScript `Promise` that has no rejection handler is rejected.

Sample code

copy

```
App({
  onUnhandledRejection(res) {
    // A JavaScript Promise that has no rejection handler is rejected.
    console.log(res.reason, res.promise);
    //res.reason describes the rejection reason and res.promise
describes the rejected Promise.
  },
});
```

Global Data

Global data can be configured in `App()`. Other pages can get and modify the global data directly.

Sample code

copy

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

FAQ

Q: Can Mini Program be closed in app.js?

A: No, Mini Program can only be closed by clicking close button in the top right corner.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_app_register-mini-program

---

Release Custom Component {#release-custom-component}

Path: miniprogram_gcash

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_custom-component_release-custom-component

---

Release Custom Component {#release-custom-component}

Last updated: 2022-07-03

Path: miniprogram_gcash

Release Custom Component

2022-07-03 18:44

Mini program natively supports the introduction of third-party npm module, so the customized component also supports publishing to npm for developers to reuse and share.

Customized Component Directory Recommended for Publishing

The following directory structure is for reference only.

File Structure

copy

```
├── src // used for individually customized component
│   ├── index.js
```

```
│   ├── index.json
│   ├── index.axml
│   └── index.acss
├── ├── demo //used for demo of customized component
│   ├── ├── index.js
│   ├── ├── index.json
│   ├── ├── index.axml
│   ├── └── index.acss
├── app.js // used for demo of customized component Mini Program
├── app.json
└── app.acss
```

JSON Sample

copy

```
// package.json
{
  "name": "your-custom-component",
  "version": "1.0.0",
  "description": "your-custom-component",
  "repository": {
    "type": "git",
    "url": "your-custom-component-repository-url"
  },
  "files": [\
    "es"\
  ],
  "keywords": [\
    "custom-component",\
    "mini-program"\
  ],
  "devDependencies": {
    "rc-tools": "6.x"
  },
  "scripts": {
    "build": "rc-tools run compile && node scripts/cp.js && node
scripts/rm.js",
    "pub": "git push origin && npm run build && npm publish"
  }
}
```

js File Sample

copy

```
// scripts/cp.js
const fs = require('fs-extra');
const path = require('path');
// copy file
fs.copySync(path.join(__dirname, '../src'), path.join(__dirname,
```

```
'../es'), {
  filter(src, des){
    return !src.endsWith('.js');
  }
});
```

copy

```
// scripts/rm.js
const fs = require('fs-extra');
const path = require('path');
// remove unnecessary file
const dirs = fs.readdirSync(path.join(__dirname, '../es'));
dirs.forEach((item) => {
  if (item.includes('app.') || item.includes('DS_Store') ||
item.includes('demo')) {
    fs.removeSync(path.join(__dirname, '../es/', item));
  } else {
    const moduleDirs = fs.readdirSync(path.join(__dirname, '../es/',
item));
    moduleDirs.forEach((item2) => {
      if (item2.includes('demo')) {
        fs.removeSync(path.join(__dirname, '../es/', item, item2));
      }
    });
  }
});
fs.removeSync(path.join(__dirname, '../lib/'));
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_custom-component_release-custom-component

---

Release Mini Program {#release-mini-program}

Last updated: 2022-07-03

Path: miniprogram_gcash

Release Mini Program

2022-07-03 18:44

Till now, the Mini Program being developed can be run inside the IDE Simulator. Only after the release, the Mini Program will be available on the AppContainer-integrated mobile app.

Create Mini Program

Before the release, the Mini Program should be created in the Mini Program Development Platform. If you are a Mini Program admin, you can create a new Mini Program on the platform. If you are a normal developer, you need to ask the Mini Program admin to add you to the group of the newly created Mini Program.

Mini Program admin can create a new Mini Program in the Mini Program Development Platform.

Add Member in Mini Program

If you are a normal Mini Program developer, please contact your Mini Program admin to add you into the members of the Mini Program. And if you are a Mini Program admin, please add the developers in the specific Mini Program.

Log into IDE

Make sure the IDE is in the login page. Click the login button at the upper-right corner to show the login dialog and fill in your account to log in.

Preview

The preview function allows the developer to preview the Mini Program in an actual device.

The preview function requires login via QR code and selection of associated application.

Upload

When the Mini Program is ready to upload, click the upper-right corner to upload them. After confirmation, the codes are uploaded to the Mini Program platform. Now a development version Mini Program is generated in the platform. If you click upload several times, a new version Mini Program will be generated. Note that the newer version does not overwrite the older one.

Submit for Reviewing

Only Mini Program admin can submit for the reviewing, the admin can log into the platform, find the version that developer has uploaded, and click the apply button to apply for review. And then wait for the reviewing result of the Mini Program.

Release Mini Program

When the review process of the Mini Program is complete, Mini Program admin would be able to select either perform a staged release or a full release to push the Mini Program to production.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/quick-start_release-mini-program

---

Release Mini Program {#release-mini-program}

Last updated: 2021-05-09

Path: miniprogram_gcash

Release Mini Program

2021-05-09 18:43

Till now, the Mini Program being developed can be run inside the IDE Simulator. Only after the release, the Mini Program will be available on the AppContainer-integrated mobile app.

Create Mini Program

Before the release, the Mini Program should be created in the Mini Program Development Platform. If you are a Mini Program admin, you can create a new Mini Program on the platform. If you are a normal developer, you need to ask the Mini Program admin to add you to the group of the newly created Mini Program.

Mini Program admin can create a new Mini Program in the Mini Program Development Platform.

Add Member in Mini Program

If you are a normal Mini Program developer, please contact your Mini Program admin to add you into the members of the Mini Program. And if you are a Mini Program admin, please add the developers in the specific Mini Program.

Log into IDE

Make sure the IDE is in the login page. Click the login button at the upper-right corner to show the login dialog and fill in your account to log in.

Preview

The preview function allows the developer to preview the Mini Program in an actual device.

The preview function requires login via QR code and selection of associated application.

Upload

When the Mini Program is ready to upload, click the upper-right corner to upload them. After confirmation, the codes are uploaded to the Mini Program platform. Now a development version Mini Program is generated in the platform. If you click upload several times, a new version Mini Program will be generated. Note that the newer version does not overwrite the older one.

Submit for Reviewing

Only Mini Program admin can submit for the reviewing, the admin can log into the platform, find the version that developer has uploaded, and click the apply button to apply for review. And then wait for the reviewing result of the Mini Program.

Release Mini Program

When the review process of the Mini Program is complete, Mini Program admin would be able to select either perform a staged release or a full release to push the Mini Program to production.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/quick-start_release-mini-program

---

Release mini programs {#release-mini-programs}

Last updated: 2022-11-13

Path: miniprogram_gcash

Release mini programs

2022-11-13 15:01

This topic describes the steps of the task to publish a mini program. When the mini program is ready to be released, developer admins can apply for publishing the mini program.

Procedures

To release a version, you can follow the corresponding steps as below:

Step 1: Version created

After uploading a version from Mini Program Studio (IDE) to the workspace, click **Mini Program** on the left menu panel and go to the **Versions** page of a mini program. You can release the version to different apps:

- **The current app** is the app where you create the mini program. The current app is added to the **App Manage** automatically after creating a workspace.
- **Target apps** are added by the wallet in the **App Manage**.

The current app

You can release the mini program to the current app in two ways:

- Choose App
- Release

Choose App

To release the mini program to different environments of the current app or other target apps configured by the wallet, Click **Choose App**.

Select the current app and choose an environment. Then click **Select** to trigger the release process.

> **Note**: You cannot release a mini program to the same environment repeatedly.

Release

To directly release the mini program to the production environment, click **Release** to trigger the release process.

Target apps

To release the version to target apps, click **Choose App** to continue.

Select a target app you want and an environment. Then click **Select** to trigger the release process.

> **Note**: You cannot release a mini program to the same environment repeatedly.

Whether to release the version to the current app or target apps, the release process continues with package building automatically. You can check and modify the configurations at this step by clicking **View Configuration**.

Then you can click **Apply to Release** to check the basic information, server domain whitelist, and H5 domain whitelist. Currently, only the basic information is supported to modify in the release process. After confirming the basic information, click **Apply**.

Step 2: Under review

The release request of the version is sent to the wallet for approval.

Step 3: Pilot testing

After the request is approved, you can set the tester whitelist to conduct the pilot testing.

Click **Set Test Whitelist** to add tester emails and then click **Add**.

During the pilot testing, you can choose to add or delete testers to update the tester whitelist.

After the pilot testing is finished, click **Finish Pilot Testing** and then click **Complete** to enter the grayscale release.

Step 4: Grayscale release (Optional)

Click **Confirm** to trigger the grayscale release.

The grayscale ratio range is from 1%-100%. You can choose to conduct the grayscale by selecting any of ratios within the range. For example, if you want to release the mini program at 1% grayscale, select 1% and click **Confirm** to trigger the process. You can find problems and make adjustments at the initial grayscale.

If everything goes well, you can gradually increase the ratio to 100% to fully release the version. Then the whole release process comes to the end.

Step 5: Final release

If the version is ready to run online after the pilot testing, click **Final Release** to fully release the version directly. And the whole release process comes to the end at this step.

Now you have completed releasing a mini program version.

Next steps

Generate QR codes

Remove mini programs

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/platform/release

Remote Debugging {#remote-debugging}

Last updated: 2021-05-09

Path: miniprogram_gcash

Remote Debugging

2021-05-09 18:43

For ease of real machine debugging, the Mini Program Studio provides the remote real machine debugging function. With the remote real machine debugging, you can:

- Perform breakpoint debugging of remote Mini Program in IDE
- View the AXML structure and style of remote interface in IDE
- View cellphone's network, storage and other information in IDE
- View Mini Program running log on cellphone in IDE

Click Debug in the top-right toolbar and confirm to push and generate debugging QR code:

After scanning with the app, the simulator shows the connection information. Meanwhile the cellphone shows the remote debugging mode has been connected. Now you can open the `DevTool` window to debug.

For example, you can normally inspect axml elements.

And you can perform break point debugging. Just select the `Sources` tab of the devtool, and then choose the specific js file. You can simple click the line number to add break point or right click the line number to show the break point prompt and then add a conditional break point. If the break point hits, the break pint line will become blue and there will be a hint in the phone showing break point hits.

Attention in the remote debugging: make sure to disconnect remote debugging after modifying code each time, and then push again, repeat the steps to scan to connect and perform remote debugging.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/mini-program-studio_debugging_remote-debugging

---

Remote Debugging {#remote-debugging}

Last updated: 2022-07-03

Path: miniprogram_gcash

Remote Debugging

2022-07-03 18:44

For ease of real machine debugging, the Mini Program Studio provides the remote real machine debugging function. With the remote real machine debugging, you can:

- Perform breakpoint debugging of remote Mini Program in IDE
- View the AXML structure and style of remote interface in IDE
- View cellphone's network, storage and other information in IDE
- View Mini Program running log on cellphone in IDE

Click Debug in the top-right toolbar and confirm to push and generate debugging QR code:

After scanning with the app, the simulator shows the connection information. Meanwhile the cellphone shows the remote debugging mode has been connected. Now you can open the `DevTool` window to debug.

For example, you can normally inspect axml elements.

And you can perform break point debugging. Just select the `Sources` tab of the devtool, and then choose the specific js file. You can simple click the line number to add break point or right click the line number to show the break point prompt and then add a conditional break point. If the break point hits, the break pint line will become blue and there will be a hint in the phone showing break point hits.

Attention in the remote debugging: make sure to disconnect remote debugging after modifying code each time, and then push again, repeat the steps to scan to connect and perform remote debugging.

九色鹿

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/mini-program-studio_debugging_remote-debugging

---

Remove mini programs {#remove-mini-programs}

Last updated: 2022-07-07

Path: miniprogram_gcash

Remove mini programs

2022-07-07 17:08

Overview

This topic provides steps for merchants to remove mini programs. To remove a mini program means that the mini program will be offline but still reserve its services in the workspace. If you want to stop running a mini program online, you can remove it.

Procedures

To remove a mini program, you can follow the corresponding steps as below:

Step 1: Navigate to mini program list

Click **Mini Program** on the menu panel to the left and choose the mini program from the list.

Step 2: Remove a mini program

Choose the mini program you want to remove and confirm its details. Then click **Apply for Removal**.

Click **Remove** to send the removal request to the wallet for review.

You can check the approval status by clicking **View Removal Application** under the **Versions** tab.

Once the request is approved, you can click **Remove Mini Program** to remove it.

You can then go to the mini program list page to check that the mini program is marked as **Archived**.

Now you have completed removing a mini program.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/platform/remove

---

SearchBar {#searchbar}

Last updated: 2022-07-03

Path: miniprogram_gcash

SearchBar

2022-07-03 18:44

The search function allows text query for the users. On basis of the current page contents, the user can perform exact search or fuzzy search to filter and locate contents and increase productivity in queries. When the search bar is activated, the cancel button appears. Note: For the purpose of UI presentation only. No service logic function is available.

Sample Code

copy

```
// API-DEMO page/component/search-bar/search-bar.json
{
  "defaultTitle": "Mini Program AntUI component library",
  "usingComponents": {
    "search-bar": "mini-antui/es/search-bar/index"
  }
}
```

copy

```
<!-- API-DEMO page/component/search-bar/search-bar.axml -->
<view>
  <search-bar
    value="{{value}}"
    placeholder="Search "
    onInput="handleInput"
    onClear="handleClear"
    onFocus="handleFocus"
    onBlur="handleBlur"
    onCancel="handleCancel"
    onSubmit="handleSubmit"
    showCancelButton="{{false}}" />
</view>
```

copy

```
// API-DEMO page/component/search-bar/search-bar.js
Page({
  data: {
    value: 'Food',
  },
  handleInput(value) {
    this.setData({
      value,
    });
  },
  handleClear(value) {
    this.setData({
      value: '',
    });
  },
```

```
    handleFocus() {},
    handleBlur() {},
    handleCancel() {
      this.setData({
        value: '',
      });
    },
    handleSubmit(value) {
      my.alert({
        content: value,
      });
    },
  });
```

Attributes

| | | | | | | --- | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Default** | **Required** | | value | Current value in search box. | String | - | No | | placeholder | Placeholder. | String | - | No | | focus | Get cursor automatically. | Boolean | false | No | | onInput | Trigger on keyboard input. | (value: String) => void | - | No | | onClear | Trigger on clicking clear icon. | (val: String) => void | - | No | | onFocus | Trigger on getting focus. | () => void | - | No | | onBlur | Trigger on losing focus. | () => void | - | No | | onCancel | Trigger on clicking cancel. | () => void | - | No | | onSubmit | Trigger on clicking enter on button. | (val: String) => void | - | No | | disabled | Set disabled. | Boolean | - | No | | maxLength | Maximum number of characters allowed for input Number. | - | No | | | showCancelButton | Always show cancel button or not. | Boolean | - | No |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_form_searchbar

---

SelectorQuery Overview {#selectorquery-overview}

Last updated: 2021-05-09

Path: miniprogram_gcash

SelectorQuery Overview

2021-05-09 18:43

The class of selector query object.

Functions

| | | | --- | --- | | **Name** | **Description** | | SelectorQuery.boundingClientRect | Put the location of current selected node into the query result. | | SelectorQuery.exec | Put the query result into the `Callback`. | | SelectorQuery.scrollOffset | Put the scroll of current selected node into the query result. | | SelectorQuery.select | Select the first matched node. | | SelectorQuery.selectAll | Select all the matched nodes. | | SelectorQuery.selectViewport | The instance of select window. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_ui_selector-query_query_selectorquery-overview

---

SelectorQuery.boundingClientRect {#selectorqueryboundingclientrect}

Last updated: 2021-05-09

Path: miniprogram_gcash

SelectorQuery.boundingClientRect

2021-05-09 18:43

Put the location of the current selected node into the query result. It is similar to the `getBoundingClientRect` of `DOM`, the returned value includes width, height, left, top, bottom, right. If current node is window object, only width and height will be returned.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_ui_selector-query_query_selectorquery-boundingclientrect

---

SelectorQuery.exec {#selectorqueryexec}

Last updated: 2021-05-09

Path: miniprogram_gcash

SelectorQuery.exec

2021-05-09 18:43

Put the query result into callback function. The query result is an array according to the query sequence, the object in the array is the result of each query. If the selected node is the list of node, the query result of the single query is also an array.

Note: The `exec` should be invoked after the `onReady` of the page.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_ui_selector-query_query_selectorquery-exec

---

SelectorQuery.scrollOffset {#selectorqueryscrolloffset}

Last updated: 2021-05-09

Path: miniprogram_gcash

SelectorQuery.scrollOffset

2021-05-09 18:43

Put the scroll information of current selected node into the query result, the returned value includes scrollTop, scrollLeft.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_ui_selector-query_query_selectorquery-scrolloffset

---

SelectorQuery.select {#selectorqueryselect}

Last updated: 2021-05-09

Path: miniprogram_gcash

SelectorQuery.select

2021-05-09 18:43

Select the first node that matches the selector, the selector can support ID selector and class selector.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_ui_selector-query_query_selectorquery-select

---

SelectorQuery.selectAll {#selectorqueryselectall}

Last updated: 2021-05-09

Path: miniprogram_gcash

SelectorQuery.selectAll

2021-05-09 18:43

Select all the nodes that match the selector, the selector can support ID selector and class selector.

九色鹿

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_ui_selector-query_query_selectorquery-selectall

---

SelectorQuery.selectViewport {#selectorqueryselectviewport}

Last updated: 2021-05-10

Path: miniprogram_gcash

SelectorQuery.selectViewport

2021-05-10 03:43

The object of select window.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_ui_selector-query_query_selectorquery-selectviewport

---

Settings {#settings}

Last updated: 2022-07-03

Path: miniprogram_gcash

Settings

2022-07-03 18:44

Click the settings icon in the bottom-left corner to enter the settings interface. The settings mainly contains following ways:

- Global settings or workspace settings about the editor and other coding related settings.

- Shortcuts settings.
- Appearance settings such as theme of color and icon.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/mini-program-studio_settings

---

Settings {#settings}

Last updated: 2022-07-07

Path: miniprogram_gcash

Settings

2022-07-07 17:08

The **Settings** functionality enables workspace admins to customize the platform according to different business requirements. Other member roles can only set the two-factor authentication with this functionality.

Features

As a workspace admin, you can change the following settings:

- **Logo & Favicon**

Change the logo displayed on your customized Mini Program platform and console, and the favicon displayed on the web browser tab.

- **Service Mail Preference**

Set your noreply email, which enables you to send outgoing emails that do not accept replies. This prevents your email inbox from being overloaded with replies.

- **Domain Preference**

Customize your domain name.

- **Whitelist for Mini Program Testers**（Optional）

Workspace admins enable gray-box testing for designated testers, then developers can add tester accounts to the whitelist under **Mini Program > Configuration > Whitelist for Mini Program Testers**.

Notes:

- The maximum number of the tester accounts is 100.
- To use this feature, the Griver and WallerAPI version in the App Container must meet the following requirements:
- Griver 2.16.0 or higher
- WalletAPI 0.4.4 or higher
- **Two-Factor Authentication**

You can set up the 2FA to to add an extra layer of security for your Mini Program platform account. For more information, see Set Two-Factor Authentication on how to enable this feature.

- **URL of About Us**

Customize a link for your **About Us** page. By the clickable text on your portal homepage, you can link the Mini Program platform with one of your websites.

- **Term of Service Agreement**

You can upload the agreement. By the clickable text on your portal homepage, your users can preview or download the agreement file.

- **Copyright Notice**

You can enter the texts in the format of a symbol, a year and an owner to define your copyright notice, which is displayed on the portal homepage.

More information

Overview

Member Role

Workflow Procedures

Manage Mini Program

Manage Workspace

Authorization

Approvals

Manage Apps

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/platform/setting

---

Simulator {#simulator}

Last updated: 2022-07-03

Path: miniprogram_gcash

Simulator

2022-07-03 18:44

After the Mini Program project builds, it will run in the simulator automatically. You can click and slide in the screen to simulate the click and slide motion in real device.

By default, each time saving the changes of the code, the simulator will refresh automatically to achieve real-time update. If you want to disable the feature, cancel the auto refresh selection in the bottom of the simulator.

The top of the simulator window mainly contains following functions:

- Device switch: choose different size devices including iOS and Android. You can also create a custom device.
- Scale control: control the scale of the Mini Program.
- Refresh: compile the project and refresh the simulator.
- Tools: tools for simulation data such as you can mock the location.
- Simulation log: check the compile logs.
- Standalone window: set the simulator to a standalone window.

The bottom of the simulator window mainly contains following functions:

- Page path: show current page path. Click the path, the relative `.js` file will be opened automatically.
- Page params: show the parameters of current page.
- Auto refresh: checkbox for auto refresh of simulator.

Device Switch

The developer can select different devices or add custom device to debug the adaptation problem of Mini Program on the models of different sizes.

Scale Control

The developer can scale the display of the simulator via preset percentages.

Simulation Tools

The simulation tools is a useful function for developers. Click the tools menu, you can display or hide the tools panel.

Home

Click the `Home` button, the Mini Program will go to background, which can be used to test `onShow` and `onHide` function in `app.js` or `page.js`.

Location

Click the `Location` button, you can mock the location of simulator. Then `my.getLocation` will return the mocked data.

> Note:
>
> Float data is required when inputting the longitude and latitude.

Scan

Mock the `my.scan` API, you can input the scan result, then in Mini Program, `my.scan` will get the mocked data.

Shake

Simulate the shake of the device, used to test `my.watchShake` API.

Corp

Simulate the capture screen event of users, used to test `my.onUserCaptureScreen` API.

MemWarn

Simulate the memory warning event of app, used to test `my.onMemoryWarning` API.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/mini-program-studio_interface_simulator-interface

---

Simulator {#simulator}

Last updated: 2021-05-09

Path: miniprogram_gcash

Simulator

2021-05-09 18:43

After the Mini Program project builds, it will run in the simulator automatically. You can click and slide in the screen to simulate the click and slide motion in real device.

By default, each time saving the changes of the code, the simulator will refresh automatically to achieve real-time update. If you want to disable the feature, cancel the auto refresh selection in the bottom of the simulator.

The top of the simulator window mainly contains following functions:

- Device switch: choose different size devices including iOS and Android. You can also create a custom device.
- Scale control: control the scale of the Mini Program.
- Refresh: compile the project and refresh the simulator.
- Tools: tools for simulation data such as you can mock the location.
- Simulation log: check the compile logs.
- Standalone window: set the simulator to a standalone window.

The bottom of the simulator window mainly contains following functions:

- Page path: show current page path. Click the path, the relative `.js` file will be opened automatically.
- Page params: show the parameters of current page.
- Auto refresh: checkbox for auto refresh of simulator.

Device Switch

The developer can select different devices or add custom device to debug the adaptation problem of Mini Program on the models of different sizes.

Scale Control

The developer can scale the display of the simulator via preset percentages.

Simulation Tools

The simulation tools is a useful function for developers. Click the tools menu, you can display or hide the tools panel.

Home

Click the `Home` button, the Mini Program will go to background, which can be used to test `onShow` and `onHide` function in `app.js` or `page.js`.

Location

Click the `Location` button, you can mock the location of simulator. Then `my.getLocation` will return the mocked data.

> Note:
>
> Float data is required when inputting the longitude and latitude.

Scan

Mock the `my.scan` API, you can input the scan result, then in Mini Program, `my.scan` will get the mocked data.

Shake

Simulate the shake of the device, used to test `my.watchShake` API.

Corp

Simulate the capture screen event of users, used to test `my.onUserCaptureScreen` API.

MemWarn

Simulate the memory warning event of app, used to test `my.onMemoryWarning` API.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/mini-program-studio_interface_simulator-interface

---

Stepper {#stepper}

Last updated: 2022-07-03

Path: miniprogram_gcash

Stepper

2022-07-03 18:44

Increase or decrease the current value.

Note:

• No prompt for input of maximum. If it exceeds the maximum, the system automatically displays the value as the maximum.

• Input of decimal is not supported. It is possible to use + and - to change value.

Sample Code

copy

```
// API-DEMO page/component/stepper/stepper.json
{
  "defaultTitle": "Stepper",
  "usingComponents":{
    "stepper": "mini-antui/es/stepper/index",
    "list": "mini-antui/es/list/index",
    "list-item": "mini-antui/es/list/list-item/index"
  }
}
```

copy

```
<!-- API-DEMO page/component/stepper/stepper.axml -->
<list>
  <list-item disabled="{{true}}">
    Show number value
    <view slot="extra">
      <stepper onChange="callBackFn" step="{{1}}" showNumber
readOnly="{{false}}" value="{{value}}" min="{{2}}" max="{{12}}" />
    </view>
  </list-item>
  <list-item disabled="{{true}}">
    Do not show number value
    <view slot="extra">
      <stepper onChange="callBackFn" step="{{1}}" readOnly="
{{false}}" value="{{value}}" min="{{2}}" max="{{12}}" />
    </view>
  </list-item>
  <list-item disabled="{{true}}">
    Disabled
    <view slot="extra">
      <stepper onChange="callBackFn" showNumber value="{{11}}"
min="{{2}}" max="{{12}}" disabled />
    </view>
  </list-item>
  <list-item disabled="{{true}}">
    readOnly
    <view slot="extra">
      <stepper onChange="callBackFn" showNumber value="{{11}}"
min="{{2}}" max="{{12}}" readOnly />
```

```
        </view>
      </list-item>
      <list-item>
        <button onTap="modifyValue">Modify stepper initial
value</button>
      </list-item>
    </list>
```

copy

```
    // API-DEMO page/component/stepper/stepper.js
    Page({
      data: {
        value: 8,
      },
      callBackFn(value){
       console.log(value);
      },
      modifyValue() {
        this.setData({
          value: this.data.value + 1,
        });
      }
    });
```

Attributes

| | | | | | | --- | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Default** | **Required** | | min | Minimum. | Number | 0 | Yes | | max | Maximum. | Number | 10000 | Yes | | value | Initial value. | Number | 10 | Yes | | step | Change step, can be a decimal. | Number | 1 | No | | onChange | Change callback function. | (value: Number) => void | - | No | | disabled | Disabled. | Boolean | false | No | | readOnly | Input read-only. | Boolean | false | No | | showNumber | Show number or not, not shown by default. | Boolean | false | No |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_others_stepper

---

Steps {#steps}

Last updated: 2022-07-03

Path: miniprogram_gcash

Steps

2022-07-03 18:44

Show the progress bar as per the steps.

| | | | | | | --- | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Default** | **Required** | | className | Outermost layer overlapping style. | String | | No | | activeIndex | Current active step. | Number | 1 | Yes | | failIndex | Current failed step (effective only in vertical mode). | Number | 0 | No | | direction | Displaying direction, options including `vertical` and `horizontal`. | String | horizontal | No | | size | Uniform icon size, in px. | Number | 0 | No | | items | Step details. | Array[{title, description, icon, activeIcon, size}] | [] | Yes |

Items attribute detailed description

| | | | | | | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Required** | | items.title | Title of step details. | String | Yes | | items.description | Description of step details. | String | Yes | | items.icon | Icon for unreached step (effective only in vertical mode). | String | Yes | | items.activeIcon | Icon for reached step (effective only in vertical mode) | String | Yes | | items.size | Size of icon for reached step, in px. (effective only in vertical mode) | Number | Yes |

Example

copy

```
{
  "usingComponents": {
    "steps": "mini-antui/es/steps/index"
  }
}
```

copy

```
<steps
  activeIndex="{{activeIndex}}"
  items="{{items}}"
</steps>
```

copy

```
Page({
  data: {
    activeIndex: 1,
    items: [{\
      title: 'Step one',\
      description: 'This is step one',\
    }, {\
      title: 'Step two',\
      description: 'This is step two',\
    }, {\
      title: 'Step three',\
      description: 'This is step three',\
    }]
```

```
  }
});
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_layout-navigation_steps

---

Suggestions on Performance Optimization {#suggestions-on-performance-optimization}

Last updated: 2021-05-09

Path: miniprogram_gcash

Suggestions on Performance Optimization

2021-05-09 18:43

Operating Principle

Different from the traditional H5 applications, Mini Program operation architecture is divided into two parts -- webview and worker. The webview is for rendering, and the worker is for data storage and service logic execution.

1. Communication between webview and worker is asynchronous. This means the data is not rendered immediately when setData is called, and asynchronous transmission from worker to webview occurs.

2. During the transmission, the data is serialized as a string, and transferred by means of evaluateJavascript. The data size affects the performance.

Optimizing First Screen

The first screen may be defined differently. Here it means the first meaningful render of the service. For example: with regard to a list page, the first screen means the contents rendered for the first time in the list.

Controlling Size of Mini Program Resource Package

When the use accesses Mini Program for the first time, mobile App client downloads Mini Program resource package from CDN, so the size of the resource package affects the Mini Program startup performance.

Optimization suggestions

- Delete the useless image resources, because all image resources are packaged by default.
- Control the size of images and avoid using large picture. It is recommended to upload large pictures via CDN channels.
- Clear useless codes in time

Advance Data Request to onLoad

- Upon operation, Mini Program triggers the onLoad lifecycle function of the page, and then transfers the initial page data from worker to webview for the initial render.
- When the initial page render is completed, a notification is sent from webview to worker and triggers the onReady lifecycle function.

Some Mini Programs send requests in onReady which causes delay of first screen render.

Optimization suggestion

Advance data request to onLoad

Control the Number of Nodes to Be Rendered at Once in the First Screen

After the service request is returned, it generally calls the setData to trigger page re-render. The execution process is as below:

1. Data is sent from worker to webview
2. webview constructs virtual DOM as per the data transferred, makes difference comparison with the previous data (starting from the root node), and starts render.

Due to the data serialization in communication from worker to webview, and then the execution of evaluateJavascript in the webview, the first screen render performance is affected if the data transmitted once is too large.

in addition, if the construction nodes are too many or the nested hierarchy is too deep on webview, say, more than 100 list items to be rendered once in the list page of some Mini Program and each list item containing nested contents, but less than 10 items to be displayed on the whole screen, the different comparison takes long time, a large number of DOMs are constructed once in the first screen, and the first screen render performance is compromised.

Optimization suggestions

- setData data quantity should not be too large; do not transfer too long list once.
- Do not construct too many nodes on the first screen. The service end may request a large quantity of data once. Do not run setData all at once. It is possible to setData partial data and wait for a while (say, 400ms, depending on the specific service) and then call $spliceData to transfer the remaining data.

Optimize setData Logic

Any page change triggers setData. At the same time, multiple setData may trigger the page re-render. The following four interfaces trigger webview page re-render.

- **Page.prototype.setData:** Triggers the difference comparison of the whole page
- **Page.prototype.$spliceData:** Optimizes long list and avoid transferring whole list all at once and triggering the difference comparison of the whole page
- **Component.prototype.setData:** Starts the difference comparison from the corresponding component node
- **Component.prototype.$spliceData:** Optimizes long list and avoid transferring whole list all at once. Only makes difference comparison from the corresponding component node.

Optimization suggestions

- Avoid triggering setData or $spliceData frequently, no matter on the page level or component level. In our analyzed cases, some pages contain countdown logic but the countdown is triggered too frequently (in microseconds).
- When it is required to trigger re-render frequently, avoid using page-level setData or $spliceData. This block can be encapsulated into a custom component, and then the component-level setData and $spliceData can be used to trigger component re-render.
- For render of long data list, use $spliceData to append data in several times instead of transfer of the whole list.
- For complicated page, it is recommended to encapsulate it into custom component to minimize the page-level setData.

Optimization case

Suggest specifying path to set data:

copy

```
this.setData({
  'array[0]': 1,
  'obj.x':2,
});
```

Not suggesting the following method (although this.data is copied, the attribute is changed directly):

copy

```
const array = this.data.array.concat();
array[0] = 1;
const obj={...this.data.obj};
obj.x=2;
this.setData({array,obj});
```

Even not suggesting direct change of this.data (violating the immutable data principle):

copy

```
this.data.array[0]=1;
this.data.obj.x=2;
this.setData(this.data)
```

Using $spliceData for long list

copy

```
this.$spliceData({ 'a.b': [1, 0, 5, 6] })
```

Note:

Sometimes when service logic are encapsulated in component, it is only required to call setData within the component when the component UI needs re-render. In other occasions, however, it is required to trigger component re-render from the page. For example, the onPageScroll event is monitored on page, and it is required to notify the corresponding component to render again when the event is trigger. Now the measure is as below:

copy

```
// /pages/index/index.js
Page({
    onPageScroll(e) {
        if (this.xxcomponent) {
            this.xxcomponent.setData({
                scrollTop: e.scrollTop
            })
        }
    }
})
// /components/index/index.js
Component({
    didMount(){
        this.$page.xxcomponent = this;
    }
})
```

It is possible to mount the component to the corresponding page in the didMount, so that the call of component-level setData in the page triggers re-render of the component only.

Use Key Parameter

The “key” can be used in “for” to increase performance. Note that the “key” cannot be set on blocks.

Sample codes:

copy

```
<view a:for="{{array}}" key="{{item.id}}"></view>
<block a:for="{{array}}"><view key="{{item.id}}"></view></block>
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_suggestions-on-performance-optimization

---

Suggestions on Performance Optimization {#suggestions-on-performance-optimization}

Last updated: 2022-07-03

Path: miniprogram_gcash

Suggestions on Performance Optimization

2022-07-03 18:44

Operating Principle

Different from the traditional H5 applications, Mini Program operation architecture is divided into two parts -- webview and worker. The webview is for rendering, and the worker is for data storage and service logic execution.

1. Communication between webview and worker is asynchronous. This means the data is not rendered immediately when setData is called, and asynchronous transmission from worker to webview occurs.

2. During the transmission, the data is serialized as a string, and transferred by means of evaluateJavascript. The data size affects the performance.

Optimizing First Screen

The first screen may be defined differently. Here it means the first meaningful render of the service. For example: with regard to a list page, the first screen means the contents rendered for the first time in the list.

Controlling Size of Mini Program Resource Package

When the use accesses Mini Program for the first time, mobile App client downloads Mini Program resource package from CDN, so the size of the resource package affects the Mini Program startup performance.

Optimization suggestions

- Delete the useless image resources, because all image resources are packaged by default.
- Control the size of images and avoid using large picture. It is recommended to upload large pictures via CDN channels.
- Clear useless codes in time

Advance Data Request to onLoad

- Upon operation, Mini Program triggers the onLoad lifecycle function of the page, and then transfers the initial page data from worker to webview for the initial render.
- When the initial page render is completed, a notification is sent from webview to worker and triggers the onReady lifecycle function.

Some Mini Programs send requests in onReady which causes delay of first screen render.

Optimization suggestion

Advance data request to onLoad

Control the Number of Nodes to Be Rendered at Once in the First Screen

After the service request is returned, it generally calls the setData to trigger page re-render. The execution process is as below:

1. Data is sent from worker to webview
2. webview constructs virtual DOM as per the data transferred, makes difference comparison with the previous data (starting from the root node), and starts render.

Due to the data serialization in communication from worker to webview, and then the execution of evaluateJavascript in the webview, the first screen render performance is affected if the data transmitted once is too large.

in addition, if the construction nodes are too many or the nested hierarchy is too deep on webview, say, more than 100 list items to be rendered once in the list page of some Mini Program and each list item containing nested contents, but less than 10 items to be displayed on the whole screen, the different comparison takes long time, a large number of DOMs are constructed once in the first screen, and the first screen render performance is compromised.

Optimization suggestions

- setData data quantity should not be too large; do not transfer too long list once.
- Do not construct too many nodes on the first screen. The service end may request a large quantity of data once. Do not run setData all at once. It is possible to setData partial data and wait for a while (say, 400ms, depending on the specific service) and then call $spliceData to transfer the remaining data.

Optimize setData Logic

Any page change triggers setData. At the same time, multiple setData may trigger the page re-render. The following four interfaces trigger webview page re-render.

- **Page.prototype.setData:** Triggers the difference comparison of the whole page
- **Page.prototype.$spliceData:** Optimizes long list and avoid transferring whole list all at once and triggering the difference comparison of the whole page
- **Component.prototype.setData:** Starts the difference comparison from the corresponding component node
- **Component.prototype.$spliceData:** Optimizes long list and avoid transferring whole list all at once. Only makes difference comparison from the corresponding component node.

Optimization suggestions

- Avoid triggering setData or $spliceData frequently, no matter on the page level or component level. In our analyzed cases, some pages contain countdown logic but the countdown is triggered too frequently (in microseconds).
- When it is required to trigger re-render frequently, avoid using page-level setData or $spliceData. This block can be encapsulated into a custom component, and then the component-level setData and $spliceData can be used to trigger component re-render.
- For render of long data list, use $spliceData to append data in several times instead of transfer of the whole list.
- For complicated page, it is recommended to encapsulate it into custom component to minimize the page-level setData.

Optimization case

Suggest specifying path to set data:

copy

```
this.setData({
  'array[0]': 1,
  'obj.x':2,
});
```

Not suggesting the following method (although this.data is copied, the attribute is changed directly):

copy

```
const array = this.data.array.concat();
array[0] = 1;
const obj={...this.data.obj};
obj.x=2;
this.setData({array,obj});
```

Even not suggesting direct change of this.data (violating the immutable data principle):

copy

```
this.data.array[0]=1;
this.data.obj.x=2;
this.setData(this.data)
```

Using $spliceData for long list

copy

```
this.$spliceData({ 'a.b': [1, 0, 5, 6] })
```

Note:

Sometimes when service logic are encapsulated in component, it is only required to call setData within the component when the component UI needs re-render. In other occasions, however, it is required to trigger component re-render from the page. For example, the onPageScroll event is monitored on page, and it is required to notify the corresponding component to render again when the event is trigger. Now the measure is as below:

copy

```
// /pages/index/index.js
Page({
    onPageScroll(e) {
        if (this.xxcomponent) {
            this.xxcomponent.setData({
                scrollTop: e.scrollTop
            })
        }
    }
})
// /components/index/index.js
Component({
    didMount(){
        this.$page.xxcomponent = this;
    }
})
```

It is possible to mount the component to the corresponding page in the didMount, so that the call of component-level setData in the page triggers re-render of the component only.

Use Key Parameter

The “key” can be used in “for” to increase performance. Note that the “key” cannot be set on blocks.

Sample codes:

copy

```
<view a:for="{{array}}" key="{{item.id}}"></view>
<block a:for="{{array}}"><view key="{{item.id}}"></view></block>
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_suggestions-on-performance-optimization

---

SwipeAction {#swipeaction}

Last updated: 2022-07-03

Path: miniprogram_gcash

SwipeAction

2022-07-03 18:44

Sliding cell

Sample Code

copy

```
// API-DEMO page/component/swiper-action/swiper-action.json
{
  "defaultTitle": "SwipeAction",
  "usingComponents": {
    "list": "mini-antui/es/list/index",
    "list-item": "mini-antui/es/list/list-item/index",
    "swipe-action": "mini-antui/es/swipe-action/index"
  }
}
```

copy

```
<!-- API-DEMO page/component/swiper-action/swiper-action.axml -->
<view>
  <list>
    <view a:for="{{list}}" key="{{item.content}}">
      <swipe-action
        index="{{index}}"
        restore="{{swipeIndex === null || swipeIndex !== index}}"
        right="{{item.right}}"
        onRightItemClick="onRightItemClick"
        onSwipeStart="onSwipeStart"
        extra="item{{index}}"
      >
```

```
            <list-item
              arrow="horizontal"
              index="{{index}}"
              key="items-{{index}}"
              onClick="onItemClick"
              last="{{index === list.length - 1}}"
            >
              {{item.content}}
            </list-item>
          </swipe-action>
        </view>
      </list>
    </view>
```

copy

```
    // API-DEMO page/component/swiper-action/swiper-action.js
    Page({
      data: {
        swipeIndex: null,
        list: [\
          { right: [{ type: 'edit', text: ' Unfavorite ', bgColor:
'#ccc', fColor: '#f00' }, { type: 'delete', text: ' Delete ', bgColor:
'#0ff', fColor: '#333' }], content: ' Text & background color change
at the same time Execute swipe deletion recovery ' },\
          { right: [{ type: 'delete', text: ' Delete ' }], content:
'AAA' },\
          { right: [{ type: 'edit', text: ' Unfavorite ' }, { type:
'delete', text: ' Delete ' }], content: 'BBB' },\
          { right: [{ type: 'delete', text: ' Delete ' }], content:
'CCC' },\
        ],
      },
      onRightItemClick(e) {
        const { type } = e.detail;
        my.confirm({
          title: 'Tips',
          content:
`${e.index}-${e.extra}-${JSON.stringify(e.detail)}`,
          confirmButtonText: 'Confirm',
          cancelButtonText: 'Cancel',
          success: (result) => {
            const { list } = this.data;
            if (result.confirm) {
              if (type === 'delete') {
                list.splice(this.data.swipeIndex, 1);
                this.setData({
                  list: [...list],
                });
              }

              my.showToast({
```

```
                content: 'Confirm => Execute swipe deletion recovery
',
              });
              e.done();
            } else {
              my.showToast({
                content: 'Cancel => Swipe deletion status remains
unchanged ',
              });
            }
          },
        });
      },
      onItemClick(e) {
        my.alert({
          content: `dada${e.index}`,
        });
      },
      onSwipeStart(e) {
        this.setData({
          swipeIndex: e.index,
        });
      },
    });
```

Attributes

| | | | | | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Default** | | className | Customized class. | String | - | | right | Sliding option, at most two options. | Array[Object{type: edit/delete, text: string, fColor: 'Color value', bgColor: 'Color value'}] | [] | | onRightItemClick | Click sliding option. | ({index, detail, extra, done}) => void | Call done to fold swipeAction | | restore | Restore the component to its initial status. When there are multiple swipeAction components, to slide one of them, it is required to set the restore attribute of the others as true, which prevents multiple swipeAction becomes active on the same page. | Boolean | false | | onSwipeStart | Start sliding callback. | (e: Object) => void | - | | extra | Extra information, to get in the onRightItemClick callback. | any | - |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_gesture_swipeaction

---

TabBar FAQ {#tabbar-faq}

Last updated: 2021-05-09

Path: miniprogram_gcash

TabBar FAQ

2021-05-09 18:43

Supported Function FAQ

Q: Does the page of tab bar support redirecting with parameters?

A: Yes, the page of tab bar support jumping with parameters.

Q: Does the location of tab bar support to be set to the top?

A: The location of tab bar does not support custom settings now.

Q: How to monitor tab bar tapping event?

A: You can monitor tab bar tapping event by using `onTabItemTap` in Mini Program.

Q: Does the icon of tab bar support SVG format?

A: SVG format is not supported, only PNG/JPEG/JPG/GIF format are supported.

Q: How to set the style of tab bar?

A: You can set the style of tab bar in the JSON, which is shown as follows. And you can also call my.setTabBarStyle to set.

copy

```
"tabBar": {
    "textColor": "#404040",
    "selectedColor": "#108ee9",
    "backgroundColor": "#F5F5F9"
}
```

Exception Requests FAQ

Q: What to do if "Cannot read property `getCurrentPages` of undefined" is reported when switching the tab bar?

A: Error path. Please check the path of tab bar.

Q: Why tab bar is not displayed after the page is redirected?

A: If the user enter the page by my.navigateTo or my.redirectTo, the bottom tab bar is not displayed. The first page of tab bar must be the homepage.

Q: How to obtain the upper page path after entering the page of tab bar?

A: Save the current page path globally when entering the tab bar page, and you can get the upper page path by using the global address when switching tab bar pages.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_ui_tabbar_tabbarfaq

---

TabBar FAQ {#tabbar-faq}

Last updated: 2022-07-03

Path: miniprogram_gcash

TabBar FAQ

2022-07-03 18:44

Supported Function FAQ

Q: Does the page of tab bar support redirecting with parameters?

A: Yes, the page of tab bar support jumping with parameters.

Q: Does the location of tab bar support to be set to the top?

A: The location of tab bar does not support custom settings now.

Q: How to monitor tab bar tapping event?

A: You can monitor tab bar tapping event by using `onTabItemTap` in Mini Program.

Q: Does the icon of tab bar support SVG format?

A: SVG format is not supported, only PNG/JPEG/JPG/GIF format are supported.

Q: How to set the style of tab bar?

A: You can set the style of tab bar in the JSON, which is shown as follows. And you can also call my.setTabBarStyle to set.

copy

```
"tabBar": {
    "textColor": "#404040",
    "selectedColor": "#108ee9",
    "backgroundColor": "#F5F5F9"
}
```

Exception Requests FAQ

Q: What to do if "Cannot read property `getCurrentPages` of undefined" is reported when switching the tab bar?

A: Error path. Please check the path of tab bar.

Q: Why tab bar is not displayed after the page is redirected?

A: If the user enter the page by my.navigateTo or my.redirectTo, the bottom tab bar is not displayed. The first page of tab bar must be the homepage.

Q: How to obtain the upper page path after entering the page of tab bar?

A: Save the current page path globally when entering the tab bar page, and you can get the upper page path by using the global address when switching tab bar pages.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_UI_TabBar_TabbarFAQ

Tabs {#tabs}

Last updated: 2022-07-03

Path: miniprogram_gcash

Tabs

2022-07-03 18:44

Tabs allow the user to switch between different views.

Tabs

| | | | | | | --- | --- | --- | --- | --- | | **Property** | **Type** | **Default** | **Required** | **Description** | | className | String | | No | Customized class. | | activeCls | String | | | Customized class for activating tabbar. | | tabs | Array | | Yes | tab data, including the tab `title`. The badge type `badgeType` includes `dot` and `text`, and is not displayed if the `badgeType` is not set. Badge text `badgeText` takes effect when the `badgeType` is `text`. | | activeTab | Number | | Yes | Index of the currently active tab. | | showPlus | Boolean | false | No | Show the “+” icon or not. | | onPlusClick | () => {} | | No | Callback when the “+” icon is clicked. | | onTabClick | (index: Number) => void | | No | Callback when the tab is clicked. | | onChange | (index: Number) => void | | No | Triggered when tab changes. | | swipeable | Boolean | true | No | If it is possible to switch contents by swiping. | | duration | Number | 500(ms) | No | Duration of wiping animation in ms, when the swipeable is true. | | tabBarBackgroundColor | String | | No | tabBar background color. | | tabBarActiveTextColor | String | | No | Active Tab text color of the tabBar. | | tabBarInactiveTextColor | String | | No | Inactive Tab text color of the tabBar. | | tabBarUnderlineColor | String | | No | tabBar underline color. | | tabBarCls | String | | No | tabBar custom style class. |

Tab-content

View content

| | | | | --- | --- | --- | | **Property** | **Description** | **Type** | | index | Unique index of list item. | String |

Example

copy

```
{
  "defaultTitle": "AntUI Component Library",
  "usingComponents": {
    "tabs": "mini-antui/es/tabs/index",
    "tab-content": "mini-antui/es/tabs/tab-content/index"
  }
}
```

copy

```
<view>
  <tabs
    tabs="{{tabs}}"
    showPlus="{{true}}"
    onTabClick="handleTabClick"
    onChange="handleTabChange"
    onPlusClick="handlePlusClick"
```

```
    activeTab="{{activeTab}}"
  >
    <block a:for="{{tabs}}">
      <tab-content key="{{index}}">
        <view class="tab-content">content of {{item.title}}</view>
      </tab-content>
    </block>
  </tabs>
</view>
```

copy

```
Page({
  data: {
    tabs: [\
      {\
        title: 'Option',\
        badgeType: 'text',\
        badgeText: '6',\
      },\
      {\
        title: 'Option two',\
        badgeType: 'dot',\
      },\
      { title: '3 Tab' },\
      { title: '4 Tab' },\
      { title: '5 Tab' },\
    ],
    activeTab: 2,
  },
  handleTabClick({ index }) {
    this.setData({
      activeTab: index,
    });
  },
  handleTabChange({ index }) {
    this.setData({
      activeTab: index,
    });
  },
  handlePlusClick() {
    my.alert({
      content: 'plus clicked',
    });
  },
});
```

copy

```
.tab-content {
  display: flex;
  justify-content: center;
```

```
  align-items: center;
  height: 300px;
}
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_layout-navigation_tabs

---

Tag {#tag}

Last updated: 2022-07-03

Path: miniprogram_gcash

Tag

2022-07-03 18:44

You can use the tag component to highlight the information, such as the warning.

Sample code

See the sample codes in different languages:

.json

copy

```
{
 "defaultTitle": "Tag",
 "usingComponents": {
   "tag": "mini-ali-ui/es/tag/index",
   "list-item": "mini-ali-ui/es/list/list-item/index",
   "am-switch": "mini-ali-ui/es/am-switch/index"
 }
}
```

.axml

copy

```
<view style="padding: 12px;">

  <view style="display: flex; justify-content: space-evenly;">

    <tag size="lg" iconType="{{useIcon ? 'qr' : ''}}" ghost="
```

```
{{ghost}}" type="primary">tag</tag>

    <tag size="lg" iconType="{{useIcon ? 'qr' : ''}}" ghost="
{{ghost}}" type="warning">tag</tag>

    <tag size="lg" iconType="{{useIcon ? 'qr' : ''}}" ghost="
{{ghost}}" type="danger">tag</tag>

    <tag size="lg" iconType="{{useIcon ? 'qr' : ''}}" ghost="
{{ghost}}" type="success">tag</tag>

  </view>

  <view style="display: flex; justify-content: space-evenly; margin-
top: 20px;">

    <tag size="sm" iconType="{{useIcon ? 'qr' : ''}}" ghost="
{{ghost}}" type="primary">tag</tag>

    <tag size="sm" iconType="{{useIcon ? 'qr' : ''}}" ghost="
{{ghost}}" type="warning">tag</tag>

    <tag size="sm" iconType="{{useIcon ? 'qr' : ''}}" ghost="
{{ghost}}" type="danger">tag</tag>

    <tag size="sm" iconType="{{useIcon ? 'qr' : ''}}" ghost="
{{ghost}}" type="success">tag</tag>

  </view>

  <view style="padding: 20px 10px;">

    <list-item>

      icon

      <am-switch slot="extra" onChange="setInfo" data-name="useIcon"
checked="{{useIcon}}"/>

    </list-item>

    <list-item>

      the style of the wireframe

      <am-switch slot="extra" onChange="setInfo" data-name="ghost"
checked="{{ghost}}"/>

    </list-item>

  </view>
```

```
</view>
```

.js

copy

```
 Page({

  data: {},

  onLoad() {},

  setInfo(e) {

    const { dataset } = e.target;

    const { name } = dataset;

    this.setData({

      [name]: e.detail.value,

    });

  },

});
```

Parameters

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | className | String | Class name. | | type | String | Tag type. Valid values are:
- `primary`
- `success`
- `warning`
- `danger`
The default value is `primary`. | | iconType | String | Icon type. The icon is a thumbnail image in the tag. | | size | String | Tag size. Valid values are:
- `lg`: large
- `sm`: small
The default value is `lg`. | | ghost | Boolean | An indicator of whether the tag has a frame. The default value is `false`. |

slot

| | | --- | --- | | **Name** | **Description** | | extra | The slot that is used to display texts in the tag. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_prompt-guide_tag

---

Template {#template}

Last updated: 2022-07-03

Path: miniprogram_gcash

Template

2022-07-03 18:44

The axml provides `template`, where the code snippet can be defined for invoking elsewhere.

It is recommended to use `template` to introduce `template` snippet because `template` specifies the action scope and uses only the `data` imported. If the `data` in the `template` does not change, the UI of the snippet will not be re-rendered.

Define Template

Use the name attribute to declare template name and then define code snippet within `<template/>`.

copy

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

Use Template

Use the **is** attribute to declare the required template and then introduce the required **data**. For example:

copy

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

copy

```
Page({
  data: {
    item: {
      index: 0,
      msg: 'this is a template',
      time: '2019-04-19',
    },
  },
});
```

The `is` attribute allows using the Mustache syntax to decide dynamically which template to render.

copy

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

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

Template Action Scope

The template has an action scope and can use the data introduced by "data". Except for the data directly introduced by "data", it is possible to use the onXX event to bind page logic for function handling. Below are the sample codes:

copy

```
<!-- templ.axml -->
<template name="msgItem">
    <view>
        <view>
            <text> {{index}}: {{msg}} </text>
            <text> Time: {{time}} </text>
        </view>
        <button onTap="onClickButton">onTap</button>
    </view>
</template>
```

copy

```
<!-- index.axml -->
<import src="./templ.axml"/>
<template is="msgItem" data="{{...item}}"/>
```

copy

```
Page({
  data: {
    item: {
      index: 0,
      msg: 'this is a template',
      time: '2019-04-22'
    }
  },
  onClickButton(e) {
    console.log('button clicked', e)
  },
});
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_axml-reference_template

---

Template {#template}

Last updated: 2021-05-09

Path: miniprogram_gcash

Template

2021-05-09 18:43

The axml provides `template`, where the code snippet can be defined for invoking elsewhere.

It is recommended to use `template` to introduce `template` snippet because `template` specifies the action scope and uses only the `data` imported. If the `data` in the `template` does not change, the UI of the snippet will not be re-rendered.

Define Template

Use the name attribute to declare template name and then define code snippet within `<template/>`.

copy

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

Use Template

Use the **is** attribute to declare the required template and then introduce the required **data**. For example:

copy

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

copy

```
Page({
  data: {
    item: {
      index: 0,
      msg: 'this is a template',
      time: '2019-04-19',
    },
  },
});
```

The `is` attribute allows using the Mustache syntax to decide dynamically which template to render.

copy

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

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

Template Action Scope

The template has an action scope and can use the data introduced by "data". Except for the data directly introduced by "data", it is possible to use the onXX event to bind page logic for function handling. Below are the sample codes:

copy

```
<!-- templ.axml -->
<template name="msgItem">
    <view>
        <view>
            <text> {{index}}: {{msg}} </text>
            <text> Time: {{time}} </text>
        </view>
        <button onTap="onClickButton">onTap</button>
    </view>
</template>
```

copy

```
<!-- index.axml -->
<import src="./templ.axml"/>
<template is="msgItem" data="{{...item}}"/>
```

copy

```
Page({
  data: {
    item: {
      index: 0,
      msg: 'this is a template',
      time: '2019-04-22'
    }
  },
  onClickButton(e) {
    console.log('button clicked', e)
  },
});
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_axml-reference_template

---

Template and Style {#template-and-style}

Last updated: 2022-07-03

Path: miniprogram_gcash

Template and Style

2022-07-03 18:44

Similar to page, custom component has its own axml template and acss style.

axml

The axml is the mandate part of custom component.

Note:

Different from page, user's customized event shall be placed in methods.

Example:

copy

```
<!-- /components/xx/index.axml -->
<view onTap="onMyClick" id="c-{{$id}}"/>
```

copy

```
Component({
  methods: {
    onMyClick(e) {
      console.log(this.is, this.$id);
    },
  },
});
```

slot

By supporting props in component js, the custom component can interact with external caller, accepting the data transferred from the external caller, calling the function transferred from the external caller, and notifying the internal change of the component to the external caller.

However, this is not enough, because the custom component is not flexible enough. In addition to data processing and notification, the Mini Program provides slot, so that the custom component axml structure can be assembled by using the axml transferred from the external caller. The external caller can transfer axml to custom component, which the custom component uses to assemble the final component axml structure.

Default slot

Sample code:

copy

```
<!-- /components/xx/index.axml -->
<view>
  <slot>
    <view>default slot & default value</view>
  </slot>
  <view>other</view>
</view>
```

Caller does not transfer axml

copy

```
<!-- /pages/index/index.axml -->
<xx />
```

Page output:

copy

```
default slot & default value
other
```

Caller transfers axml

copy

```
<!-- /pages/index/index.axml -->
<xx>
  <view>xx</view>
  <view>yy</view>
</xx>
```

Page output:

copy

```
xx
yy
other
```

The “slot” can be interpreted as the slot. The “default slot” is the default slot. If the caller does not transfer axml in the component tag , the default slot is rendered. If the caller transfers axml in the component tag , it is used to replace the default slot and assemble the final axml for render.

Named slot

The default slot can transfer one set of axml. For complicated component, it is required to render different axml at different locations, that is, to transfer multiple axml. Here it needs named slot .

Sample code:

copy

```
<!-- /components/xx/index.axml -->
<view>
  <slot>
    <view>default slot & default value</view>
  </slot>
  <slot name="header"/>
  <view>body</view>
  <slot name="footer"/>
</view>
```

Transfer only named slot

copy

```
<!-- /pages/index/index.axml -->
<xx>
  <view slot="header">header</view>
  <view slot="footer">footer</view>
</xx>
```

Page output

copy

```
default slot & default value
header
body
footer
```

Transfer named slot and default slot

copy

```
<!-- /pages/index/index.axml -->
<xx>
  <view>this is to default slot</view>
  <view slot="header">header</view>
  <view slot="footer">footer</view>
</xx>
```

Page output

copy

```
this is to default slot
header
body
footer
```

The named slot is the slot with a name. In the sub-tag of the custom component tag, the external caller can specify which part of axml to place in which named slot of the custom component. The part without named slot specified in the sub-tag of the custom component tag is placed into the default slot. If it transfers only the named slot, the default slot will not be overwritten.

slot-scope

Through the named slot, the custom component axml uses either the custom component axml, or the external caller (such as page) axml.

By using the custom component axml, it is possible to access the data within the component. Through the props attribute, meanwhile, it is possible to access the data of external caller.

Example:

copy

```
// /components/xx/index.js
Component({
  data: {
    x: 1,
  },
  props: {
    y: '',
  },
});
```

copy

```
<!-- /components/xx/index.axml -->
<view>component data: {{x}}</view>
<view>page data: {{y}}</view>
```

copy

```
// /pages/index/index.js
Page({
  data: { y: 2 },
});
```

copy

```
<!-- /pages/index/index.axml -->
<xx y="{{y}}" />
```

Page output:

copy

```
component data: 1
page data: 2
```

When the custom component uses external caller (such as page) axml through slot, it can access the data of external caller only.

Sample code:

copy

```
<!-- /components/xx/index.axml -->
<view>
  <slot>
    <view>default slot & default value</view>
  </slot>
  <view>body</view>
</view>
```

copy

```
// /pages/index/index.js
Page({
  data: { y: 2 },
});
```

copy

```
<!-- /pages/index/index.axml -->
<xx>
  <view>page data: {{y}}</view>
</xx>
```

Page output:

copy

```
page data: 2
```

The slot scope allows the slot content can access the data within the component.

Sample code:

copy

```
// /components/xx/index.js
Component({
  data: {
    x: 1,
  },
});
```

copy

```
<!-- /components/xx/index.axml -->
<view>
  <slot x="{{x}}">
    <view>default slot & default value</view>
  </slot>
  <view>body</view>
</view>
```

copy

```
// /pages/index/index.js
Page({
  data: { y: 2 },
});
```

copy

```
<!-- /pages/index/index.axml -->
<xx>
  <view slot-scope="props">
    <view>component data: {{props.x}}</view>
    <view>page data: {{y}}</view>
  </view>
</xx>
```

Page output:

copy

```
component data: 1
page data: 2
body
```

As shown above, the custom component exposes the internal component data by defining the slot attribute. When the page uses the component, the action scope slot is declared via slot-scope. The attribute value defines the temporary variable name props, thus accessible to the internal data of the component.

acss

Just like the page, the custom component can have its defined own acss style. The acss is automatically introduced into the page that uses the component without manual introduction of the page.

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_custom-component_create-custom-component_template-and-style

---

Template and Style {#template-and-style}

Last updated: 2021-05-09

Path: miniprogram_gcash

Template and Style

2021-05-09 18:43

Similar to page, custom component has its own axml template and acss style.

axml

The axml is the mandate part of custom component.

Note:

Different from page, user's customized event shall be placed in methods.

Example:

copy

```
<!-- /components/xx/index.axml -->
<view onTap="onMyClick" id="c-{{$id}}"/>
```

copy

```
Component({
  methods: {
    onMyClick(e) {
      console.log(this.is, this.$id);
    },
  },
});
```

slot

By supporting props in component js, the custom component can interact with external caller, accepting the data transferred from the external caller, calling the function transferred from the external caller, and notifying the internal change of the component to the external caller.

However, this is not enough, because the custom component is not flexible enough. In addition to data processing and notification, the Mini Program provides slot, so that the custom component axml structure can be assembled by using the axml transferred from the external caller. The external caller can transfer axml to custom component, which the custom component uses to assemble the final component axml structure.

Default slot

Sample code:

copy

```
<!-- /components/xx/index.axml -->
<view>
  <slot>
    <view>default slot & default value</view>
  </slot>
```

```
  <view>other</view>
</view>
```

Caller does not transfer axml

copy

```
<!-- /pages/index/index.axml -->
<xx />
```

Page output:

copy

```
default slot & default value
other
```

Caller transfers axml

copy

```
<!-- /pages/index/index.axml -->
<xx>
  <view>xx</view>
  <view>yy</view>
</xx>
```

Page output:

copy

```
xx
yy
other
```

The “slot” can be interpreted as the slot. The “default slot” is the default slot. If the caller does not transfer axml in the component tag , the default slot is rendered. If the caller transfers axml in the component tag , it is used to replace the default slot and assemble the final axml for render.

Named slot

The default slot can transfer one set of axml. For complicated component, it is required to render different axml at different locations, that is, to transfer multiple axml. Here it needs named slot .

Sample code:

copy

```
<!-- /components/xx/index.axml -->
<view>
  <slot>
    <view>default slot & default value</view>
```

```
  </slot>
  <slot name="header"/>
  <view>body</view>
  <slot name="footer"/>
</view>
```

Transfer only named slot

copy

```
<!-- /pages/index/index.axml -->
<xx>
  <view slot="header">header</view>
  <view slot="footer">footer</view>
</xx>
```

Page output

copy

```
default slot & default value
header
body
footer
```

Transfer named slot and default slot

copy

```
<!-- /pages/index/index.axml -->
<xx>
  <view>this is to default slot</view>
  <view slot="header">header</view>
  <view slot="footer">footer</view>
</xx>
```

Page output

copy

```
this is to default slot
header
body
footer
```

The named slot is the slot with a name. In the sub-tag of the custom component tag, the external caller can specify which part of axml to place in which named slot of the custom component. The part without named slot specified in the sub-tag of the custom component tag is placed into the default slot. If it transfers only the named slot, the default slot will not be overwritten.

slot-scope

Through the named slot, the custom component axml uses either the custom component axml, or the external caller (such as page) axml.

By using the custom component axml, it is possible to access the data within the component. Through the props attribute, meanwhile, it is possible to access the data of external caller.

Example:

copy

```
// /components/xx/index.js
Component({
  data: {
    x: 1,
  },
  props: {
    y: '',
  },
});
```

copy

```
<!-- /components/xx/index.axml -->
<view>component data: {{x}}</view>
<view>page data: {{y}}</view>
```

copy

```
// /pages/index/index.js
Page({
  data: { y: 2 },
});
```

copy

```
<!-- /pages/index/index.axml -->
<xx y="{{y}}" />
```

Page output:

copy

```
component data: 1
page data: 2
```

When the custom component uses external caller (such as page) axml through slot, it can access the data of external caller only.

Sample code:

copy

```
<!-- /components/xx/index.axml -->
<view>
  <slot>
    <view>default slot & default value</view>
  </slot>
  <view>body</view>
</view>
```

copy

```
// /pages/index/index.js
Page({
  data: { y: 2 },
});
```

copy

```
<!-- /pages/index/index.axml -->
<xx>
  <view>page data: {{y}}</view>
</xx>
```

Page output:

copy

```
page data: 2
```

The slot scope allows the slot content can access the data within the component.

Sample code:

copy

```
// /components/xx/index.js
Component({
  data: {
    x: 1,
  },
});
```

copy

```
<!-- /components/xx/index.axml -->
<view>
  <slot x="{{x}}">
    <view>default slot & default value</view>
  </slot>
  <view>body</view>
</view>
```

copy

```
// /pages/index/index.js
Page({
  data: { y: 2 },
});
```

copy

```
<!-- /pages/index/index.axml -->
<xx>
  <view slot-scope="props">
    <view>component data: {{props.x}}</view>
    <view>page data: {{y}}</view>
  </view>
</xx>
```

Page output:

copy

```
component data: 1
page data: 2
body
```

As shown above, the custom component exposes the internal component data by defining the slot attribute. When the page uses the component, the action scope slot is declared via slot-scope. The attribute value defines the temporary variable name props, thus accessible to the internal data of the component.

acss

Just like the page, the custom component can have its defined own acss style. The acss is automatically introduced into the page that uses the component without manual introduction of the page.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_custom-component_create-custom-component_template-and-style

---

Terms {#terms}

Last updated: 2022-07-03

Path: miniprogram_gcash

Terms

2022-07-03 18:44

You can use the terms component when users must agree with terms before using or activating the service. Normally a link to the terms is provided for user's reference.

Sample code

See the sample codes in different languages:

.json

copy

```
{
   "defaultTitle": "Terms",
   "usingComponents": {
     "terms": "mini-ali-ui/es/terms/index"
   }
}
```

.axml

copy

```
<view>
 <terms onSelect="onSelect" related="{{c1.related}}" hasDesc="
{{c1.hasDesc}}" agreeBtn="{{c1.agreeBtn}}" cancelBtn="
{{c1.cancelBtn}}">
   <view class="text" slot="header">
     <text>
       Agree
       <navigator class="link" url="https://example.com">user
authorization terms</navigator>
     </text>
   </view>
 </terms>
 <text class="title">double button</text>
</view>
<view>
 <terms onSelect="onSelect" fixed="{{c2.fixed}}" related="
{{c2.related}}" hasDesc="{{c2.hasDesc}}" agreeBtn="{{c2.agreeBtn}}"
cancelBtn="{{c2.cancelBtn}}" shape="{{c2.shape}}" capsuleMinWidth="
{{c2.capsuleMinWidth}}" capsuleSize="{{c2.capsuleSize}}">
   <view class="text" slot="desc">
     <text>
       check
       <navigator class="link" url="https://example.com">ETC Service
User Terms</navigator>
       Authorize ETC service to obtain ID card and delivery address
for ETC application. Pay attention to the owner's service life number
for approval
```

```
      </text>
    </view>
  </terms>
  <text class="title">Title with dditional description</text>
</view>
<view>
  <terms onSelect="onSelect" fixed="{{c3.fixed}}" related="
{{c3.related}}" hasDesc="{{c3.hasDesc}}" agreeBtn="{{c3.agreeBtn}}"
cancelBtn="{{c3.cancelBtn}}">
    <view class="text" slot="header">
      <text>
        agree
        <navigator class="link" url="https://example.com">User
Authorization Terms</navigator>
      </text>
    </view>
  </terms>
  <text class="title">Binding protocol is selected</text>
</view>
<view>
  <terms onSelect="onSelect" fixed="{{c4.fixed}}" related="
{{c4.related}}" hasDesc="{{c4.hasDesc}}" agreeBtn="{{c4.agreeBtn}}"
cancelBtn="{{c4.cancelBtn}}" shape="{{c4.shape}}" capsuleMinWidth="
{{c4.capsuleMinWidth}}" capsuleSize="{{c4.capsuleSize}}">
    <view class="text" slot="header">
      <text>
        agree
        <navigator class="link" url="https://example.com">User
Authorization Terms</navigator>
      </text>
    </view>
  </terms>
  <text class="title">Binding protocol is not selected</text>
</view>
<view>
  <terms fixed="{{c5.fixed}}" related="{{c5.related}}" hasDesc="
{{c5.hasDesc}}"  agreeBtn="{{c5.agreeBtn}}" cancelBtn="
{{c5.cancelBtn}}" shape="{{c5.shape}}" capsuleMinWidth="
{{c5.capsuleMinWidth}}" capsuleSize="{{c5.capsuleSize}}">
    <view class="text" slot="header">
      <text>
        agree
        <navigator class="link" url="https://example.com">User
Authorization Terms</navigator>
      </text>
    </view>
  </terms>
  <text class="title">without binding protocol</text>
</view>
<view style="padding-bottom:30px;">
  <terms fixed="{{c6.fixed}}" related="{{c6.related}}" hasDesc="
```

```
{{c6.hasDesc}}" agreeBtn="{{c6.agreeBtn}}" cancelBtn="
{{c6.cancelBtn}}" shape="{{c6.shape}}" capsuleMinWidth="
{{c6.capsuleMinWidth}}" capsuleSize="{{c6.capsuleSize}}">
   <view class="text" slot="header">
     <text>
       agree
       <navigator class="link" url="https://example.com">User
Authorization Terms</navigator>
     </text>
   </view>
 </terms>
 <text class="title">bottom suction</text>
</view>
```

.acss

copy

```
.title{
   text-align: center;
   display: block;
   width: 100%;
   margin: 20px 0;
}
page {
   padding: 24px  12px;
}
```

.js

copy

```
const cfg = {
 c1: {
   related: false,
   agreeBtn: {
     title: 'agree the term and open',
   },
   cancelBtn: {
     title: 'Not open temporarily, manual payment',
   },
   hasDesc: false,
 },
 c2: {
   related: false,
   agreeBtn: {
     title: 'agree the term and open',
   },
   hasDesc: true,
 },
 c3: {
```

```
    related: true,
    agreeBtn: {
      checked: true,
      title: 'submit',
    },
  },
  c4: {
    related: true,
    agreeBtn: {
      title: 'submit',
    },
  },
  c5: {
    related: false,
    agreeBtn: {
      title: 'agree the term and submit',
    },
  },
  c6: {
    related: true,
    fixed: true,
    agreeBtn: {
      checked: true,
      title: 'submit',
    },
  },
};
Page({
  data: cfg,
  onLoad() {
  },
  onSelect(e) {
    const selectedData = e.currentTarget.dataset.name || '';
    selectedData && my.alert({
      title: 'Terms Btns',
      content: selectedData,
    });
  },
});
```

Parameters

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | fixed | Boolean | An indicator of whether to display the terms at the bottom of the page. The default value is `false`. | | related | Boolean | An indicator of whether the user needs to select the checkbox. The default value is `true`. | | agreeBth | Object | The button that users can click to agree with the terms. The default value is `{"title":"", "subtitle":"", "type":"primary", "data":1, "checked":false }`. | | cancelBtn | Object | The button that users can click to not agree with the terms. The default value is `{"title":"", "subtitle":"", "type":"default", "data":2 }`. | | capsuleSize | String | Capsule button size. Valid

values are:
- `large`
- `medium`
- `small`
The default value is `medium`. | | shape | String | Button shape. Valid values are:
- `default`
- `capsule`
The default value is `default`. | | capsuleMinWidth | Boolean | An indicator of whether to use the minimum width for the capsule button. The default value is `false`. | | hasDesc | Boolean | An indicator of whether to display the description about the terms. The default value is `false`. | | onSelect | EventHandle | The event that is triggered when users click the agree button. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_layout_terms

---

The Main Interface {#the-main-interface}

Last updated: 2021-05-09

Path: miniprogram_gcash

The Main Interface

2021-05-09 18:43

The main interface of Mini Program Studio mainly contains following components:

- Menu bar: including files, editor, window and other basic software settings, these settings are similar with the settings of normal development software.
- Tool bar: including functions such as associating Mini Program, toggling displays, preview, remote debug, upload and other functions for Mini Program.
- Function panel: including project file management, search, git management, NPM package management and other functions.
- Editor: for Mini Program coding.
- Simulator: local simulator for previewing Mini Program and remote debugging.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/mini-program-studio_interface_main-interface

---

The Main Interface {#the-main-interface}

Last updated: 2022-07-03

Path: miniprogram_gcash

The Main Interface

2022-07-03 18:44

The main interface of Mini Program Studio mainly contains following components:

- Menu bar: including files, editor, window and other basic software settings, these settings are similar with the settings of normal development software.
- Tool bar: including functions such as associating Mini Program, toggling displays, preview, remote debug, upload and other functions for Mini Program.
- Function panel: including project file management, search, git management, NPM package management and other functions.
- Editor: for Mini Program coding.
- Simulator: local simulator for previewing Mini Program and remote debugging.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/mini-program-studio_interface_main-interface

---

Tips {#tips}

Last updated: 2022-07-03

Path: miniprogram_gcash

Tips

2022-07-03 18:44

Tool tips Including two types `tips-dialog` and `tips-plain`.

tips-dialog

| | | | | | --- | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Default** | **Required** | | className | Custom class. | String | | No | | show | Show control component or not. | Boolean | true | No | | type | `dialog` indicates the style of dialog box, `rectangle` for rectangle style. | String | dialog | No | | onCloseTap | When the `type` value is `rectangle`, component clicking close the icon callback. | () => void | | No | | iconUrl | Show the icon url. | String | | No |

Slots

| | | | --- | --- | | **slotName** | **Description** | | content | Used to render tip text contents. | | operation | Used to render right-hand operation area. |

tips-plain

| | | | | | | --- | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Default** | **Required** | | className | Custom class. | String | | No | | time | Automatic close time. (in milliseconds) | Number | 5000(ms) | No | | onClose | Callback and close tip box. | () => void | | No |

Example

copy

```
{
  "defaultTitle": "AntUI Component Library",
  "usingComponents": {
    "tips-dialog": "mini-antui/es/tips/tips-dialog/index",
    "tips-plain": "mini-antui/es/tips/tips-plain/index"
  }
}
```

tips-dialog

copy

```
<view>
  <tips-dialog
    show="{{showDialog}}"
    className="dialog"
    type="dialog"
  >
    <view class="content" slot="content">
      <view>hello,</view>
      <view>Welcome to use the Mini Program extension component
library</view>
    </view>
    <view slot="operation" class="opt-button"
onTap="onDialogTap">OK</view>
  </tips-dialog>
  <tips-dialog
    iconUrl="https://img.example.com/example.png"
    type="rectangle"
    className="rectangle"
    onCloseTap="onCloseTap"
    show="{{showRectangle}}">
    <view class="content" slot="content">
      Add to home page
```

```
    </view>
    <view slot="operation" class="add-home" onTap="onRectangleTap">Add
it now</view>
  </tips-dialog>
</view>
```

copy

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

copy

```
.rectangle {
  position: fixed;
  bottom: 100px;
}

.dialog {
  position: fixed;
  bottom: 10px;
}

.content {
  font-size: 14px;
  color: #fff;
}

.opt-button {
  width: 51px;
  height: 27px;
  display: flex;
  justify-content: center;
```

```
  align-items: center;
  color: #fff;
  font-size: 12px;
  border: #68BAF7 solid 1rpx;
}

.add-home {
  width: 72px;
  height: 27px;
  display: flex;
  justify-content: center;
  align-items: center;
  background-color: #56ADEB;
  color: #fff;
  font-size: 14px;
}
```

tips-plain

copy

```
<tips-plain onClose="onClose" time="{{time}}">{{content}}</tips-plain>
```

copy

```
Page({
  data: {
    content: 'OK',
    time: 2000,
  },
  onClose() {
    my.alert({
      title: '12321'
    });
  }
});
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_prompt-guide_tips

---

Title {#title}

Last updated: 2022-07-03

Path: miniprogram_gcash

Title

2022-07-03 18:44

You can use the title component to display the title of each page.

Sample code

See the sample codes in different languages:

.json

copy

```
{
 "defaultTitle": "title",
 "usingComponents": {
   "title": "mini-ali-ui/es/title/index"
 }
```

.axml

copy

```
<title
 hasLine="true"
 type="more"
 onActionTap="titleMore"
>Title without icon</title>
<title
 hasLine="true"
 iconURL="https://example.com/images/T1HHFgXXVeXXXXXXXX.png"
 type="close"
 onActionTap="titleClose"
>Title with a close action</title>
<title
 hasLine="true"
 className="changeColor"

iconURL="https://example.com/mdn/miniProgram_mendian/afts/img/A*wiFYTo!
 type="arrow"
 onActionTap="titleGo"
>Modify the style by class</title>
```

.acss

copy

```
.changeColor {
 font-size: 30px;
```

```
 color: #f32600;
}
```

.js

copy

```
Page({
 data: {},
 onLoad() {},
 titleGo() {
   my.showToast({
     content: 'click the arrow to jump',
   });
 },
 titleMore() {
   my.showToast({
     content: 'click the more to display bubble menu',
   });
 },
 titleClose() {
   my.showToast({
     content: 'click the close to close',
   });
 },
});
```

Parameters

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | className | String | Customized class. | | hasLine | Boolean | An indicator of whether a line is required under the title. The default value is `false`. | | iconURL | String | URL of the icon next to title name. The icon is displayed as a square image by default. | | type | String | Type of the icon that users can tap. Valid values are:
- `arrow`
- `close`
- `more`
- `custom`: The customized content is empty by default and you need to specify the slot that is named *operation*.
This property is null by default. When the property is null, *onActionTap* is invalid. | | onActionTap | EventHandle | The event that is triggered when users tap the icon on the right of title. The default value is `() => {}`. The event is valid only when *type* is specified. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_layout_title

---

Tool Bar {#tool-bar}

Last updated: 2021-05-09

Path: miniprogram_gcash

Tool Bar

2021-05-09 18:43

The tool bar of the Mini Program Studio locates in the top of the software. It contains the core function of the Mini Program Studio.

You can see the descriptions for each option in the toolbar from the left to right.

Associate Mini Program

One developer account can have multiple Mini Programs. After the account login, developers are required to associate the Mini Program under development. Associated application decides which code package will be uploaded to which Mini Program when you click to upload the codes.

Toggle Display Area

The middle of the tool bar can control whether display the coding area, devtools view and simulator.

> Note: the function panel will display or dismiss together with coding area. The coding area and devtools can not be hidden at the same time.

Compiling Mode

By default it is in normal compiling mode. In other words, the default refresh simulator will open the home page and not pass in any parameter. You may add custom compiling mode so that it starts from another page upon the simulator refresh with related parameters, which will increase debugging efficiency.

By clicking the `Compile selector` and then click the `New option`, you can create a new compiling mode.

Clear Cache

Clear the build cache and network cache.

Remote Debugging with Real Machine

During the real machine debugging, it is possible to view the debugging information in Mini Program Studio, and you can also set breakpoint, check runtime logs. For details see Remote Debugging.

Preview with Real Machine

Use app to scan the QR code and preview the Mini Program in the app of real machine. The QR code will be invalid after 15 minutes.

Upload

On basis of the associated Mini Program, the Mini Program codes are uploaded to the Mini Program Developer Portal to build an executable program in app. The uploaded version can be specified, if not specified, the current version is incremented by 1(the current version must be greater than the previous version). After the upload is completed, a unique development version is generated in the Mini Program Developer Portal.

For the version management standard and specifications, see Semver.

Details

Click the `Details` button in the tool bar, the details window will display in the editor area.

The details mainly contains following information:

- The associated Mini Program name, local project path and online version of the Mini Program.
- Modify the project configuration
- Enable component2 compile: it needs to be enabled for custom component, see details here.
- Enable Axml strict check: it will check the grammar of the axml file in strict mode, which can help to improve the quality of the code.
- Enable parallel loader: it will use multiple processes to build the project to make it faster.
- Enable distFile minify: minify the source code. By default, in preview and debug mode, the code is not minified. In production, it will always be minified.
- Ignore the domain check for request API such as `my.request`, `my.uploadFile` in simulation, preview and debug mode.
- Ignore the domain check for web-view component in simulation, preview and debug mode.

Login

If you have not login to the Mini Program Studio, click the `Login` button to login.

After login, click the avatar, you can choose to logout the Mini Program Studio. If you exit the Mini Program Studio after logout, next time when you re-open the Mini Program Studio, login is required.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/mini-program-studio_interface_tool-bar-interface

---

Tool Bar {#tool-bar}

Last updated: 2022-07-03

Path: miniprogram_gcash

Tool Bar

2022-07-03 18:44

The tool bar of the Mini Program Studio locates in the top of the software. It contains the core function of the Mini Program Studio.

You can see the descriptions for each option in the toolbar from the left to right.

Associate Mini Program

One developer account can have multiple Mini Programs. After the account login, developers are required to associate the Mini Program under development. Associated application decides which code package will be uploaded to which Mini Program when you click to upload the codes.

Toggle Display Area

The middle of the tool bar can control whether display the coding area, devtools view and simulator.

> Note: the function panel will display or dismiss together with coding area. The coding area and devtools can not be hidden at the same time.

Compiling Mode

By default it is in normal compiling mode. In other words, the default refresh simulator will open the home page and not pass in any parameter. You may add custom compiling mode so that it starts from another page upon the simulator refresh with related parameters, which will increase debugging efficiency.

By clicking the `Compile selector` and then click the `New option`, you can create a new compiling mode.

Clear Cache

Clear the build cache and network cache.

Remote Debugging with Real Machine

During the real machine debugging, it is possible to view the debugging information in Mini Program Studio, and you can also set breakpoint, check runtime logs. For details see Remote Debugging.

Preview with Real Machine

Use app to scan the QR code and preview the Mini Program in the app of real machine. The QR code will be invalid after 15 minutes.

Upload

On basis of the associated Mini Program, the Mini Program codes are uploaded to the Mini Program Developer Portal to build an executable program in app. The uploaded version can be specified, if not specified, the current version is incremented by 1(the current version must be greater than the previous version). After the upload is completed, a unique development version is generated in the Mini Program Developer Portal.

For the version management standard and specifications, see Semver.

Details

Click the `Details` button in the tool bar, the details window will display in the editor area.

The details mainly contains following information:

- The associated Mini Program name, local project path and online version of the Mini Program.
- Modify the project configuration

- Enable component2 compile: it needs to be enabled for custom component, see details here.
- Enable Axml strict check: it will check the grammar of the axml file in strict mode, which can help to improve the quality of the code.
- Enable parallel loader: it will use multiple processes to build the project to make it faster.
- Enable distFile minify: minify the source code. By default, in preview and debug mode, the code is not minified. In production, it will always be minified.
- Ignore the domain check for request API such as `my.request`, `my.uploadFile` in simulation, preview and debug mode.
- Ignore the domain check for web-view component in simulation, preview and debug mode.

Login

If you have not login to the Mini Program Studio, click the `Login` button to login.

After login, click the avatar, you can choose to logout the Mini Program Studio. If you exit the Mini Program Studio after logout, next time when you re-open the Mini Program Studio, login is required.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/mini-program-studio_interface_tool-bar-interface

---

Try Mini Program Demo {#try-mini-program-demo}

Last updated: 2022-07-03

Path: miniprogram_gcash

Try Mini Program Demo

2022-07-03 18:44

Download Mini Program Studio

Firstly, please download Mini Program Studio. It is a desktop application that helps the development of Mini Program, including local debugging, code editing, preview on device, publish and other functions that cover the whole workflow of Mini Program development.

Try Your First Mini Program

Open the IDE and click '+' to add new project.

Input the Mini Program name and select the project path, then click 'Complete' to open the project.

Preview your first Mini Program in the editor.

Now you have completed the creation of your first Mini Program project in local. Next, start your journey in the editor at the right side.

The following contents introduce in steps how to develop the demo, and get your through the basic development workflow through the demo. After the development, refer the release flow to publish the Mini Program.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/quick-start_try-mini-program-demo

---

Try Mini Program Demo {#try-mini-program-demo}

Last updated: 2021-05-09

Path: miniprogram_gcash

Try Mini Program Demo

2021-05-09 18:43

Download Mini Program Studio

Firstly, please download Mini Program Studio. It is a desktop application that helps the development of Mini Program, including local debugging, code editing, preview on device, publish and other functions that cover the whole workflow of Mini Program development.

Try Your First Mini Program

Open the IDE and click '+' to add new project.

Input the Mini Program name and select the project path, then click 'Complete' to open the project.

Preview your first Mini Program in the editor.

Now you have completed the creation of your first Mini Program project in local. Next, start your journey in the editor at the right side.

The following contents introduce in steps how to develop the demo, and get your through the basic development workflow through the demo. After the development, refer the release flow to publish the Mini Program.

九色鹿

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/quick-start_try-mini-program-demo

---

Two-factor authentication {#two-factor-authentication}

Last updated: 2022-07-07

Path: miniprogram_gcash

Two-factor authentication

2022-07-07 17:08

What's two-factor authentication?

Two-factor authentication (2FA) is an authentication method that requires two factors to verify a user's identity. Users are required to provide authentication via these following ways:

- Something you know, such as your username and password.
- Something you have, such as your mobile device.

As two types of authentication is required, the 2FA feature safeguards your account even if your password has been leaked. This extra layer of security also protects your account from malicious attacks and data breaches. The Mini Program platform provides the 2FA feature via the Google Authenticator.

Procedures

You can set the 2FA under **Settings** > **Two-Factor Authentication**. See the procedures below on how to enable and disable 2FA:

Enable the two-factor authentication

To enable the 2FA, complete the following steps:

1. Go to **Settings** > **Two-Factor Authentication** and click **Activate**.

2. Download the Google Authenticator app on your mobile device.

3. Use the Google Authenticator app to scan the QR code displayed on the Mini Program Platform. If you are not able to scan the QR code, click on **Enter a text code** and the Mini Program platform will generate a time-based key which you can enter into the app.

4. The Google Authenticator app will generate a 6-digit verification code, which you need to enter into the Mini Program platform.

Note:

If you have changed your mobile device or are not able to access it, you need to disable the 2FA and set it up again on a new device.

Disable the two-factor authentication

You can disable the 2FA any time after it's enabled. To disable the 2FA, complete the following steps:

1. Go to **Settings** > **Two-Factor Authentication** and click **Disable**.
2. The Mini Program platform will send a verification code to your account email.
3. Enter the verification code in the Mini Program platform and confirm to disable the 2FA.

More information

Settings

Manage Mini Programs

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/platform/set-2fa

---

Understand the Mini Program File Structure {#understand-the-mini-program-file-structure}

Last updated: 2022-07-03

Path: miniprogram_gcash

Understand the Mini Program File Structure

2022-07-03 18:44

Overview

This section uses Todo program demo as an example to introduce the file structure of Mini Program and the role of each file type in the Mini Program.

Directory Structure

Let's know about the overall directory structure of Mini Program from the following notes

copy

```
assets          // Store various static resources, such as images
components      // Custom component directory of Mini Program
  --add-button  // Here we defined a component called add-button
pages           // All pages included in the Mini Program are placed
under “pages”, one folder per page
  --add-todo    // Mini Program page
  --Todos       // Mini Program page
app.js          // Here some global service logic is configured for
the Mini Program, such as global method and global variables
app.acss        // Global style configuration, here the styles take
effect on every page
app.json        // Some basic configuration info for the Mini Program
pages, such as page path
```

json

The json file is used to setup Mini Program configuration. For example, the `app.json` includes the related configuration of the whole Mini Program.

copy

```
// app.json

{
  "pages": [\
    "pages/todos/todos",\
    "pages/add-todo/add-todo"\
  ],
  "window": {
    "defaultTitle": "Todo App",
    "titleBarColor": "#323239"
  }
}
```

1. The pages attribute is an array. Each string in the array defines the page path of the Mini Program. In the demo of todo list, two directories are configured externally. It is required to add these page configuration after you adding some pages.

2. The defaultTitle configuration in the window defines the title in the navigation bar of the Mini Program: "Todo App". The `titleBarColor` specifies the navigation bar's background color as hexadecimal color value.

For other configurations of the app.json file, click here.

The above-mentioned json file includes the global json configurations. Each page or component has related json configuration to specify the component dependence and so on. Click here.

axml

Generally, the axml can be regarded as the html of the Mini Program, which differs from the html in terms of:

1. Different tags. For example, Mini Program uses to replace .

2. The types of tags supported by axml are fewer than html.

3. The axml tag has its own parsing syntax to realize traverse, conditional judgment and other advanced operations. The html only includes static tags.

copy

```
<view class="todo-item {{completed ? 'checked' : ''}}">
  {{number}}
</view>
```

In axml, the format like `{{ }}` is used to render variables or execute simple operation. For example, the “completed” above is a ternary operation. When the “completed” is true, the class is rendered as “todo-item checked” or just ”todo-item”.

The `{{number}}` variable shows the results accordingly after rendering as per the assignment.

acss

The acss extends the css capability while supporting most of the css syntax. In contrast to css, the major differences are:

1. Supporting rpx unit calculation

2. Import acss in other path using `@import`

3. For more details, click here.

js

The js file is used to describe the code logic. Each page needs a js file to describe the logic of the current page. The following codes are used for the illustration simply.

copy

```
// pages/todos/todo.js
const app = getApp();
Page({
  data: {},
  onLoad() {},
  onTodoChanged(e) {}
});
```

The Page class is the constructor of the page, and should be written during the lifecycle of each page.

1. data

- The data object is considered as the axml rendering context. Simply put, if the data has a name with attribute 'Mini Program', the corresponding axml file can use the form {{name}} to read 'Mini Program'. When the setData method is used to make change in “data”, the data in axml changes in real time.

2. onLoad

- When this page is initialized at the first time by the user, this function is called.

3. onTodoChanged (user customized function)

- This is a user-customized method. The user can defines more custom functions to implement more capabilities.

In the above contents we have known about the function of each file type in the Mini Program. Next, we will explain Todo App demo in details.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/quick-start_understand-the-mini-program-file-structure

---

Understand the Mini Program File Structure {#understand-the-mini-program-file-structure}

Last updated: 2021-05-09

Path: miniprogram_gcash

Understand the Mini Program File Structure

2021-05-09 18:43

Overview

This section uses Todo program demo as an example to introduce the file structure of Mini Program and the role of each file type in the Mini Program.

Directory Structure

Let's know about the overall directory structure of Mini Program from the following notes

copy

```
assets          // Store various static resources, such as images
components      // Custom component directory of Mini Program
  --add-button  // Here we defined a component called add-button
pages           // All pages included in the Mini Program are placed
under "pages", one folder per page
  --add-todo    // Mini Program page
  --Todos       // Mini Program page
app.js          // Here some global service logic is configured for
the Mini Program, such as global method and global variables
app.acss        // Global style configuration, here the styles take
effect on every page
app.json        // Some basic configuration info for the Mini Program
pages, such as page path
```

json

The json file is used to setup Mini Program configuration. For example, the `app.json` includes the related configuration of the whole Mini Program.

copy

```
// app.json

{
  "pages": [\
    "pages/todos/todos",\
    "pages/add-todo/add-todo"\
  ],
  "window": {
    "defaultTitle": "Todo App",
    "titleBarColor": "#323239"
  }
}
```

1. The pages attribute is an array. Each string in the array defines the page path of the Mini Program. In the demo of todo list, two directories are configured externally. It is required to add these page configuration after you adding some pages.

2. The defaultTitle configuration in the window defines the title in the navigation bar of the Mini Program: "Todo App". The `titleBarColor` specifies the navigation bar's background color as hexadecimal color value.

For other configurations of the app.json file, click here.

The above-mentioned json file includes the global json configurations. Each page or component has related json configuration to specify the component dependence and so on. Click here.

axml

Generally, the axml can be regarded as the html of the Mini Program, which differs from the html in terms of:

1. Different tags. For example, Mini Program uses to replace .

2. The types of tags supported by axml are fewer than html.

3. The axml tag has its own parsing syntax to realize traverse, conditional judgment and other advanced operations. The html only includes static tags.

copy

```
<view class="todo-item {{completed ? 'checked' : ''}}">
  {{number}}
</view>
```

In axml, the format like `{{ }}` is used to render variables or execute simple operation. For example, the “completed” above is a ternary operation. When the “completed” is true, the class is rendered as “todo-item checked” or just ”todo-item”.

The `{{number}}` variable shows the results accordingly after rendering as per the assignment.

acss

The acss extends the css capability while supporting most of the css syntax. In contrast to css, the major differences are:

1. Supporting rpx unit calculation

2. Import acss in other path using `@import`

3. For more details, click here.

js

The js file is used to describe the code logic. Each page needs a js file to describe the logic of the current page. The following codes are used for the illustration simply.

copy

```
// pages/todos/todo.js
const app = getApp();
Page({
  data: {},
  onLoad() {},
  onTodoChanged(e) {}
});
```

The Page class is the constructor of the page, and should be written during the lifecycle of each page.

1. data

- The data object is considered as the axml rendering context. Simply put, if the data has a name with attribute 'Mini Program', the corresponding axml file can use the form {{name}} to read 'Mini Program'. When the setData method is used to make change in “data”, the data in axml changes in real time.

2. onLoad

- When this page is initialized at the first time by the user, this function is called.

3. onTodoChanged (user customized function)

- This is a user-customized method. The user can defines more custom functions to implement more capabilities.

In the above contents we have known about the function of each file type in the Mini Program. Next, we will explain Todo App demo in details.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/quick-start_understand-the-mini-program-file-structure

---

UpdateManager Overview {#updatemanager-overview}

Last updated: 2022-07-03

Path: miniprogram_gcash

UpdateManager Overview

2022-07-03 18:44

The UpdateManager object is used to manage the mini program updates. Call my.getUpdateManager to obtain an UpdateManager instance.

Methods

| | | | --- | --- | | **Name** | **Description** | | UpdateManager.applyUpdate | Force to restart the mini program and update to the latest version. This API is called after the `onUpdateReady` callback is received, which means that the new version of the mini program is downloaded. | | UpdateManager.onCheckForUpdate | Listen for the event that a request is sent to the server to check for updates. | | UpdateManager.onUpdateReady | Listen for the event that a newer mini program version is available. | | UpdateManager.onUpdateFailed | Listen for the event that the mini program update is failed. |

Sample code

copy

```
const updateManager = my.getUpdateManager()

updateManager.onCheckForUpdate(function (res) {
  // Callback of onCheckForUpdate
  console.log(res.hasUpdate)
})

updateManager.onUpdateReady(function () {
  my.confirm({
    title: 'Update reminder',
    content: 'The new version is ready. Do you want to restart the
mini program?',
    success: function (res) {
      if (res.confirm) {
        // The new version is downloaded. Call
UpdateManager.applyUpdate to restart and apply the new version.
        updateManager.applyUpdate()
      }
    }
  })
})

updateManager.onUpdateFailed(function () {
  // The new version of the mini program is failed to be downloaded.
})
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_update_updatemanager_overview

---

UpdateManager.applyUpdate {#updatemanagerapplyupdate}

Last updated: 2022-07-03

Path: miniprogram_gcash

UpdateManager.applyUpdate

2022-07-03 18:44

Force to restart the mini program and update to the latest version. This API is called after the `onUpdateReady` callback is received, which means that the new version of the mini program is downloaded.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_update_updatemanager_applyupdate

---

UpdateManager.onCheckForUpdate {#updatemanageroncheckforupdate}

Last updated: 2022-07-03

Path: miniprogram_gcash

UpdateManager.onCheckForUpdate

2022-07-03 18:44

Listen for the event that a request is sent to the server to check for updates. Instead of being triggered actively by the developer, checking the version update is triggered automatically during the mini program cold start.

Parameter

The parameter is in object type and has the following property:

| | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | hasUpdate | Boolean | Yes | An indicator of whether a new version is available. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_update_updatemanager_checkupdate

UpdateManager.onUpdateFailed {#updatemanageronupdatefailed}

Last updated: 2022-07-03

Path: miniprogram_gcash

UpdateManager.onUpdateFailed

2022-07-03 18:44

Listen for the event that the mini program update is failed. Instead of being triggered by the developer, the client side triggers the new version downloading actively. A callback is performed when the new version of the mini program is failed to be downloaded. The failure might be caused by the network issue.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_update_updatemanager_updatefail

---

UpdateManager.onUpdateReady {#updatemanageronupdateready}

Last updated: 2022-07-03

Path: miniprogram_gcash

UpdateManager.onUpdateReady

2022-07-03 18:44

Listen for the event that a newer mini program version is available. Instead of being triggered by the developer, the client side triggers the new version downloading actively. A callback is performed after the new version of the mini program is successfully downloaded.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_update_updatemanager_updateready

---

Upload {#upload}

Last updated: 2022-07-03

Path: miniprogram_gcash

Upload

2022-07-03 18:44

After you complete the development and testing on the simulator or a real machine, the next step is to submit the Mini Program to your workspace in the Mini Program Development Portal and proceed with the subsequent operation to release the Mini Program to users.

In the Mini Program Studio, the one-key upload function is provided:

Before uploading, pay attention to the following points:

1. Firstly, you need to associate the project to a created Mini Program.
2. Online version or current uploaded version: For easy management, each upload has an incremental version. You can define your own versioning rule.

Next, click `Upload`. As the volume of the Mini Program should not exceeds 2048 KB, if it is above the volume limit, a prompt will be displayed. We suggest you to reduce resources to reduce the volume.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/mini-program-studio_upload

---

Upload mini programs {#upload-mini-programs}

Last updated: 2022-11-13

Path: miniprogram_gcash

Upload mini programs

2022-11-13 15:01

This topic describes the steps of the task to upload a mini program. Developers develop the mini program in the Mini Program Studio (IDE) and upload it to the Mini Program Platform.

Procedures

To upload a version for a mini program, you can view the corresponding steps as follow:

Step 1: Download the IDE

Open the Mini Program Platform portal and click **Resources** to download the IDE.

Once IDE is up and running, enter the account and password that you registered on Mini Program Platform to log in to IDE.

Step 2: Upload a version

Click **Mini program** on the left navigation panel and select the Todos template. After that click **Next** to continue.

Enter the name of the mini program that was newly created in the workspace.

Click **Upload** to upload the version to the workspace.

Now you have completed uploading a version for a mini program.

Next steps

Configure mini programs

Release mini programs

Generate QR code

Remove mini programs

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/platform/upload

---

Use Custom Component {#use-custom-component}

Last updated: 2021-05-09

Path: miniprogram_gcash

Use Custom Component

2021-05-09 18:43

Use Custom Component

Note:

The event of custom component (such as onTap) is not supported by every custom component by default. It cannot be used unless the custom component itself supports clearly. For details on custom component support event, see the section of component constructor.

The use of custom component is similar to the basic component.

1. In the page json file, specify the custom component to be used.

copy

```
// /pages/index/index.json
{
  "usingComponents": {
    "customer": "/components/customer/index"
  }
}
```

2. In the page axml file, use the custom component, which is similar to the basic components.

copy

```
<!-- /pages/index/index.axml -->
<view>
  <customer />
</view>
```

Citing Custom Component:

copy

```
// page.json  Note that it is not configured in app.json
{
  "usingComponents":{
    "your-custom-component":"mini-antui/es/list/index",
    "your-custom-component2":"/components/card/index",
    "your-custom-component3":"./result/index",
    "your-custom-component4":"../result/index"
  }
}
// The project absolute path starts with /; the relative path starts
with ./ or ../; the npm path does not start with /
```

Reference Information

For installing the npm module, see framework overview npm part

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_custom-component_use-custom-component

---

Use Custom Component {#use-custom-component}

Last updated: 2022-07-03

Path: miniprogram_gcash

Use Custom Component

2022-07-03 18:44

Use Custom Component

Note:

The event of custom component (such as onTap) is not supported by every custom component by default. It cannot be used unless the custom component itself supports clearly. For details on custom component support event, see the section of component constructor.

The use of custom component is similar to the basic component.

1. In the page json file, specify the custom component to be used.

copy

```
// /pages/index/index.json
{
  "usingComponents": {
    "customer": "/components/customer/index"
  }
}
```

2. In the page axml file, use the custom component, which is similar to the basic components.

copy

```
<!-- /pages/index/index.axml -->
<view>
```

```
  <customer />
</view>
```

Citing Custom Component:

copy

```
// page.json  Note that it is not configured in app.json
{
  "usingComponents":{
    "your-custom-component":"mini-antui/es/list/index",
    "your-custom-component2":"/components/card/index",
    "your-custom-component3":"./result/index",
    "your-custom-component4":"../result/index"
  }
}
// The project absolute path starts with /; the relative path starts
with ./ or ../; the npm path does not start with /
```

Reference Information

For installing the npm module, see framework overview npm part

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_custom-component_use-custom-component

---

Use OrderStr to pay {#use-orderstr-to-pay}

Last updated: 2022-07-03

Path: miniprogram_gcash

Use OrderStr to pay

2022-07-03 18:44

Users can use the wallet app to pay for the order placed in the mini program if the wallet app provides the payment service. This document introduces the payment by calling the my.tradePay API with `orderStr`. The following two payment types are supported:

- General online payment
- Pre-authorization payment

General online payment

Users can complete a general online payment in the mini program.

User experience

To complete a payment in the mini program, users usually follow the steps below:

1. The user chooses goods in the mini program and creates an order, then presses the **Pay** button in the mini program.
2. The mini program redirects the user to the wallet app and the wallet app displays the payment page.
3. The user confirms the order information, such as the payee and amount, then clicks the **Pay** button to make the payment.
4. After confirming the payment, the wallet app displays the payment result and redirects the user back to the payment result page in the mini program.

Procedures

To use the `my.tradePay` API to initiate a payment in the mini program, mini program developers must complete the following steps:

1. Confirm that the payment service provided by the wallet app supports the payment by `orderStr` and obtain the integration guide from the wallet app.
2. Integrate the payment service at the mini program server side.
3. Create a mini program in the wallet workspace on the Mini Program Platform, or make sure that the mini program can be published in the wallet app.
4. Provide goods and payment services in the mini program.
5. Publish the mini program.

Payment process

The following figure illustrates the payment process:

Figure 1. Process flow of the general payment

The payment process contains the following steps:

1. The user places an order in the mini program (Step 1).
2. The mini program client sends a request to create the order to the mini program server by calling the my.request API (Step 1.1).
3. The mini program server creates the order and returns `orderStr` to the mini program client (Step 1.1.1 & 1.1.2).
4. The mini program client initiates the payment request by calling the my.tradePay API with `orderStr` to the wallet app (Step 1.1.2.1).
5. The wallet processes the payment request internally and the wallet app renders the cashier page (Step 1.1.2.1.1 - 1.1.2.1.3).
6. The user confirms the payment and the wallet app displays the payment result (Step 2 - 2.2).

7. The wallet app returns the payment result to the mini program client. At the same time, the wallet server returns the payment result to the mini program server (Step 2.3).
8. The mini program client displays the payment result (Step 2.3.1).

Note:

The payment flow is for reference and may vary depending on the API implementation of the wallet.

Pre-authorization payment

Pre-authorization payment is a common practice in rental and hotel industries where the user can pre-authorize a payment in advance. Unlike goods with confirmed prices, the price of some services can only be determined when the user has finished using the service. As such, service merchants can use pre-authorization payments to ensure that the order can be paid before providing the service.

Merchants can call the my.tradePay API with `orderStr` to initiate a pre-authorization payment request via the mini program. After the user has granted pre-authorization, the funds are captured by the merchant automatically after the service amount is settled.

User experience

To complete a pre-authorization payment, users usually follow the steps below:

1. The user confirms to use the service provided by the mini program.
2. The mini program redirects the user to the pre-authorization page in the wallet app.
3. The user confirms the authorization, then the wallet app redirects the user back to the mini program. The user starts to use the service.
4. After the user has finished using the service and the service fee is confirmed, the funds are deducted by the merchant automatically and the user receives a notification in the wallet app.

Procedures

To use the `my.tradePay` API to complete the pre-authorization in the mini program, mini program developers must complete the following steps:

1. Confirm that the wallet app supports the pre-authorization capability and obtain the integration guide from the wallet.
2. Integrate the payment service at the mini program server side.
3. Create a mini program in the wallet workspace on the Mini Program Platform, or make sure that the mini program can be published in the wallet app.
4. Provide the service that requires pre-authorization in the mini program.
5. Publish the mini program.

Payment process

The following figure illustrates the pre-authorization payment process:

Figure 2. Process flow of the pre-authorization payment

The payment process contains the following steps:

1. The user starts to use the service in the mini program (Step 1).
2. The mini program client sends a request to create the order to the mini program server by calling the my.request API (Step 2).
3. The mini program server creates the order and returns `orderStr` to the mini program client (Step 3 & 4).
4. The mini program client initiates the pre-authorization request by calling the my.tradePay API with `orderStr` to the wallet app (Step 1.1.2.1).
5. The wallet processes the pre-authorization request internally and the wallet app renders the pre-authorization page (Step 6 - 8).
6. The user completes the pre-authorization and the wallet app returns the pre-authorization result to the mini program client (Step 10 - 12).
7. The user starts to use the service provided by the mini program. When the user has finished using the service, the mini program client sends the payment request to the mini program server by calling the my.request API (Step 13 - 15).
8. The mini program server sends the payment request by calling the server API provided by the wallet and the wallet server returns the payment result (Step 16 & 17).

Note:

The payment flow is for reference and may vary depending on the API implementation of the wallet.

Sample code

Sample code for the `my.tradePay` API calling is as follows:

copy

```
my.tradePay({
    orderStr:
'app_id=2018112803019836&biz_content=%7B%22amount%22%3A%220.02%22%2C%2
05-29+15%3A54%3A35&version=1.0',
    success: function(res) {
        my.alert({
            content: JSON.stringify(res),
        });
    },
    fail: function(res) {
        my.alert({
            content: JSON.stringify(res),
        });
    },
});
```

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_openapi_pay-with-orderstr

---

Use OrderStr to pay in Mini Program {#use-orderstr-to-pay-in-mini-program}

Last updated: 2021-05-09

Path: miniprogram_gcash

Use OrderStr to pay in Mini Program

2021-05-09 18:43

Users can use the wallet app to pay for the order placed in the mini program if the wallet app provides the payment service. This document introduces the payment by calling the my.tradePay API with *orderStr*. Two payment types are supported:

- General online payment
- Pre-authorization payment

General payment

Users can complete a general online payment in the mini program.

User experience

To complete a payment in the mini program, users usually follow the steps below:

1. The user chooses goods in the mini program and create an order, then presses the **Pay** button in the mini program.
2. The mini program redirects the user to the wallet app and the wallet app displays the payment page.
3. The user confirms the order information, such as the payee and amount, then clicks the **Pay** button for payment.
4. After confirming the payment, the wallet app displays the payment result and redirects the user back to the payment result page in the mini program.

Procedures

To use the my.tradePay API to initiate a payment in the mini program, mini program developers complete the following steps:

1. Confirm that the payment service provided by the wallet app supports the payment by *orderStr* and obtain the integration guide from the wallet app.
2. Integrate the payment service at the mini program server side.
3. Create a mini program in the wallet workspace on the Mini Program platform, or make sure that the mini program can be published in the wallet app.
4. Provide goods and payment service in the mini program.
5. Publish the mini program.

Payment process

The following figure illustrates the payment process:

Figure 1. Process flow of the general payment

The payment process contains the following steps:

1. The user places an order in the mini program. (Step 1)
2. The mini program client sends the create order request to the mini program server by calling the my.request API. (Step 1.1)
3. The mini program server creates the order and returns *orderStr* to the mini program client. (Step 1.1.1 & 1.1.2)
4. The mini program client initiates the payment request by calling the my.tradePay API with *orderStr* to the wallet app. (Step 1.1.2.1)
5. The wallet processes the payment request internally and the wallet app renders the cashier page. (Step 1.1.2.1.1 - 1.1.2.1.3)
6. The user confirms the payment and the wallet app displays the payment result. (Step 2 - 2.2)
7. The wallet app returns the payment result to the mini program client. At the same time, the wallet server returns the payment result to the mini program server. (Step 2.3)
8. The mini program client displays the payment result. (Step 2.3.1)

Note:

The payment flow is for reference and might differ depending on the API implementation of the wallet.

Pre-authorization payment

Pre-authorization payment is a common practice in rental and hotel industries where the user can pre-authorize a payment in advance. Unlike goods with confirmed prices, the price of some services can only be determined when the user has finished using the service. As such, service merchants can use pre-authorization payments to ensure that the order can be paid before providing the service.

Merchants can call the my.tradePay API with *orderStr* to initiate a pre-authorization payment request via the mini program. After the user has granted pre-authorization, the funds are captured by the merchant automatically after the service amount is settled.

User experience

To complete a pre-authorization payment, users usually follow the steps below:

1. The user confirms to use the service provided by the mini program.

2. The mini program redirects the user to the pre-authorization page in the wallet app.

3. The user confirms the authorization, then the wallet app redirects the user back to the mini program. The user starts to use the service.

4. After the user has finished using the service and the service fee is confirmed, the funds are deducted by the merchant automatically and the user receives a notification in the wallet app.

Procedures

To use the my.tradePay API to complete the pre-authorization in the mini program, mini program developers must complete the following steps:

1. Confirm that the wallet app supports the pre-authorization capability and obtain the integration guide from the wallet.
2. Integrate the payment service at the mini program server side.
3. Create a mini program in the wallet workspace on the Mini Program platform, or make sure that the mini program can be published in the wallet app.
4. Provide the service that requires pre-authorization in the mini program.
5. Publish the mini program.

Payment process

The following figure illustrates the pre-authorization payment process:

Figure 2. Process flow of the pre-authorization payment

The payment process contains the following steps:

1. The user starts to use the service in the mini program. (Step 1)
2. The mini program client sends the create order request to the mini program server by calling the my.request API. (Step 2)
3. The mini program server creates the order and returns *orderStr* to the mini program client. (Step 3 & 4)
4. The mini program client initiates the pre-authorization request by calling the my.tradePay API with *orderStr* to the wallet app. (Step 1.1.2.1)
5. The wallet processes the pre-authorization request internally and the wallet app renders the pre-authorization page. (Step 6 - 8)
6. The user completes the pre-authorization and the wallet app returns the pre-authorization result to the mini program client. (Step 10 - 12)
7. The user starts to use the service provided by the mini program. When the user has finished using the service, the mini program client sends the payment request to the mini program server by calling the my.request API. (Step 13 - 15)

8. The mini program server sends the payment request by calling the server API provided by the wallet and the wallet server returns the payment result. (Step 16 & 17)

Note:

The payment flow is for reference and might differ depending on the API implementation of the wallet.

Sample code

Sample code for the my.tradePay API calling:

copy

```
my.tradePay({
    orderStr:
'app_id=2018112803019836&biz_content=%7B%22amount%22%3A%220.02%22%2C%2
05-29+15%3A54%3A35&version=1.0',
    success: function(res) {
        my.alert({
            content: JSON.stringify(res),
        });
    },
    fail: function(res) {
        my.alert({
            content: JSON.stringify(res),
        });
    },
});
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_openapi_pay-with-orderstr

---

Use PaymentUrl to Pay in Mini Program {#use-paymenturl-to-pay-in-mini-program}

Last updated: 2021-05-09

Path: miniprogram_gcash

Use PaymentUrl to Pay in Mini Program

2021-05-09 18:43

Users can use the wallet app to pay for the order placed in the mini program if the wallet app provides the payment service. This document introduces the payment by calling the my.tradePay API with *paymentUrl*. Both the integration processes for mini program developers and the user experience for users are similar to those of the payment by *tradeNo*.

User experience

To complete a payment in the mini program, users usually follow the steps below:

1. The user chooses goods in the mini program and create an order, then presses the **Pay** button in the mini program.
2. The mini program redirects the user to the wallet app and the wallet app displays the payment page.
3. The user confirms the order information, such as the payee and amount, then clicks the **Pay** button for payment.
4. After confirming the payment, the wallet app displays the payment result and redirects the user back to the payment result page in the mini program.

Procedures

To use the *my.tradePay* API to initiate a payment in the mini program, mini program developers must complete the following steps:

1. Confirm that the payment service provided by the wallet app supports the payment by *paymentUrl* and obtain the integration guide from the wallet app.
2. Integrate the payment service at the mini program server side.
3. Create a mini program in the wallet workspace on the Mini Program Platform, or make sure that the mini program can be published in the wallet app.
4. Provide goods and payment service in the mini program.
5. Publish the mini program.

Payment process

The following figure illustrates the payment process:

The payment process contains the following steps:

1. The user places an order in the mini program. (Step 1)
2. The mini program client sends the create order request to the mini program server by calling the my.request API. (Step 2)
3. The mini program server creates the order and sends the request to the wallet server via the server API provided by the wallet. (Step 3 & 4)
4. The wallet server creates the order and generates *paymentUrl*, and returns *paymentUrl* to the mini program server. (Step 5-7)
5. The mini program server returns *paymentUrl* to the mini program client. (Step 8)
6. The mini program client initiates the payment request by calling the my.tradePay API with *paymentUrl* to the wallet app. (Step 9)
7. The wallet app obtains the order information and renders the cashier page. (Step 10-13)

8. The user confirms the payment and the wallet processes the payment request. (Step 14-17)
9. The wallet app returns the payment result to the mini program client. At the same time, the wallet server returns the payment result to the mini program server. (Step 18 & 19)
10. The mini program client displays the payment result. (Step 20)

Note:

The payment flow is for reference and might differ depending on the API implementation of the wallet.

Sample code

Sample code for the *my.tradePay* API calling with *paymentUrl*:

copy

```
my.tradePay({
    paymentUrl: 'payment url',
    success: function(res) {
        my.alert({
            content: JSON.stringify(res),
        });
    },
    fail: function(res) {
        my.alert({
            content: JSON.stringify(res),
        });
    },
});
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_openapi_pay-with-paymenturl

---

Use PaymentUrl to pay {#use-paymenturl-to-pay}

Last updated: 2022-07-03

Path: miniprogram_gcash

Use PaymentUrl to pay

2022-07-03 18:44

Users can use the wallet app to pay for the order placed in the mini program if the wallet app provides the payment service. This document introduces the payment by calling the my.tradePay API with `paymentUrl`. Both the integration processes for mini program developers and the user experience for users are similar to those of the payment by tradeNo.

User experience

To complete a payment in the mini program, users usually follow the steps below:

1. The user chooses goods in the mini program and creates an order, then presses the **Pay** button in the mini program.
2. The mini program redirects the user to the wallet app and the wallet app displays the payment page.
3. The user confirms the order information, such as the payee and amount, then clicks the **Pay** button to make the payment.
4. After confirming the payment, the wallet app displays the payment result and redirects the user back to the payment result page in the mini program.

Procedures

To use the `my.tradePay` API to initiate a payment in the mini program, mini program developers must complete the following steps:

1. Confirm that the payment service provided by the wallet app supports the payment by `paymentUrl` and obtain the integration guide from the wallet app.
2. Integrate the payment service at the mini program server side.
3. Create a mini program in the wallet workspace on the Mini Program Platform, or make sure that the mini program can be published in the wallet app.
4. Provide goods and payment services in the mini program.
5. Publish the mini program.

Payment process

The following figure illustrates the payment process:

The payment process contains the following steps:

1. The user places an order in the mini program (Step 1).
2. The mini program client sends a request to create the order to the mini program server by calling the my.request API (Step 2).
3. The mini program server creates the order and sends the request to the wallet server via the server API provided by the wallet (Step 3 & 4).
4. The wallet server creates the order and generates `paymentUrl`, then returns `paymentUrl` to the mini program server (Step 5-7).
5. The mini program server returns `paymentUrl` to the mini program client (Step 8).
6. The mini program client initiates the payment request by calling the my.tradePay API with `paymentUrl` to the wallet app (Step 9).
7. The wallet app obtains the order information and renders the cashier page (Step 10-13).

8. The user confirms the payment and the wallet processes the payment request (Step 14-17).
9. The wallet app returns the payment result to the mini program client. At the same time, the wallet server returns the payment result to the mini program server (Step 18 & 19).
10. The mini program client displays the payment result (Step 20).

Note:

The payment flow is for reference and may vary depending on the API implementation of the wallet.

Sample code

Sample code for the `my.tradePay` API calling with `paymentUrl` is as follows:

copy

```
my.tradePay({
    paymentUrl: 'payment url',
    success: function(res) {
        my.alert({
            content: JSON.stringify(res),
        });
    },
    fail: function(res) {
        my.alert({
            content: JSON.stringify(res),
        });
    },
});
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_openapi_pay-with-paymenturl

---

Use Ref to Get Component Instance {#use-ref-to-get-component-instance}

Last updated: 2022-07-03

Path: miniprogram_gcash

Use Ref to Get Component Instance

2022-07-03 18:44

The custom component supports using ref to get custom component instance. Use my.canIUse('component2') for compatibility handling.

Note:

ref can be used as well for parent component to get children component instance.

Sample Code:

copy

```
// /pages/index/index.js
Page({
  plus() {
    this.counter.plus();
  },
  saveRef(ref) {
    this.counter = ref;
  },
})
```

copy

```
<!-- /pages/index/index.axml -->
<counter ref="saveRef" />
<button onTap="plus">+</button>
```

Note: After ref is bound to saveRef, the saveRef method is triggered on component initialization.

copy

```
// /components/counter/index.js
Component({
  data: {
    counter: 0,
  },
  methods: {
    plus() {
      this.setData({ counter: this.data.counter + 1 })
    },
  },
})
```

copy

```
<!-- /components/counter/index.axml -->
<view>{{counter}}</view>
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_custom-component_create-custom-component_use-ref-to-get-component-instance

---

Use Ref to Get Component Instance {#use-ref-to-get-component-instance}

Last updated: 2021-05-09

Path: miniprogram_gcash

Use Ref to Get Component Instance

2021-05-09 18:43

The custom component supports using ref to get custom component instance. Use my.canIUse('component2') for compatibility handling.

Note:

ref can be used as well for parent component to get children component instance.

Sample Code:

copy

```
// /pages/index/index.js
Page({
  plus() {
    this.counter.plus();
  },
  saveRef(ref) {
    this.counter = ref;
  },
})
```

copy

```
<!-- /pages/index/index.axml -->
<counter ref="saveRef" />
<button onTap="plus">+</button>
```

Note: After ref is bound to saveRef, the saveRef method is triggered on component initialization.

copy

```
// /components/counter/index.js
Component({
  data: {
    counter: 0,
  },
  methods: {
```

```
    plus() {
      this.setData({ counter: this.data.counter + 1 })
    },
  },
})
```

copy

```
<!-- /components/counter/index.axml -->
<view>{{counter}}</view>
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_custom-component_create-custom-component_use-ref-to-get-component-instance

---

Use TradeNO to pay {#use-tradeno-to-pay}

Last updated: 2022-07-03

Path: miniprogram_gcash

Use TradeNO to pay

2022-07-03 18:44

Users can use the wallet app to pay for the order placed in the mini program if the wallet app provides the payment service. This document introduces the payment by calling the my.tradePay API with `tradeNo`.

User experience

To complete a payment in the mini program, users usually follow the steps below:

1. The user chooses goods in the mini program and creates an order, then presses the **Pay** button in the mini program.
2. The mini program redirects the user to the wallet app and the wallet app displays the payment page.
3. The user confirms the order information, such as the payee and amount, then clicks the **Pay** button to make the payment.
4. After confirming the payment, the wallet app displays the payment result and redirects the user back to the payment result page in the mini program.

Procedures

To use the `my.tradePay` API to initiate a payment in the mini program, mini program developers must complete the following steps:

1. Confirm that the payment service provided by the wallet app supports the payment by `tradeNO` and obtain the integration guide from the wallet.
2. Integrate the payment service at the mini program server side.
3. Create a mini program in the wallet workspace on the Mini Program Platform, or make sure that the mini program can be published in the wallet app.
4. Provide goods and payment services in the mini program.
5. Publish the mini program.

Payment process

The following figure illustrates the payment process:

Figure 1. Payment process

The payment process contains the following steps:

1. The user places an order in the mini program (Step 1).
2. The mini program client sends a request to create the order to the mini program server by calling the my.request API (Step 2).
3. The mini program server creates the order and sends the request to the wallet server via the server API provided by the wallet (Step 3 & 4).
4. The wallet server creates the order and generates `tradeNo`, then returns `tradeNo` to the mini program server (Step 5-7).
5. The mini program server returns `tradeNo` to the mini program client (Step 8).
6. The mini program client initiates the payment request by calling the my.tradePay API with `tradeNo` to the wallet app (Step 9).
7. The wallet app obtains the order information and renders the cashier page (Step 10-12).
8. The user confirms the payment and the wallet processes the payment request (Step 13-16).
9. The wallet app returns the payment result to the mini program client. At the same time, the wallet server returns the payment result to the mini program server (Step 17 & 18).
10. The mini program client displays the payment result (Step 19).

Note:

The payment flow is for reference and may vary depending on the API implementation of the wallet.

Sample code

Sample code for the `my.tradePay` API calling is as follows:

copy

```
my.tradePay({
    tradeNO: '20171115210011041053366779 2',
    success: function(res) {
        my.alert({
            content: JSON.stringify(res),
```

```
        });
    },
    fail: function(res) {
        my.alert({
            content: JSON.stringify(res),
        });
    },
});
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_openapi_pay-with-tradeno

---

Use TradeNO to pay in Mini Program {#use-tradeno-to-pay-in-mini-program}

Last updated: 2021-05-09

Path: miniprogram_gcash

Use TradeNO to pay in Mini Program

2021-05-09 18:43

Users can use the wallet app to pay for the order placed in the mini program if the wallet app provides the payment service. This document introduces the payment by calling the my.tradePay API with *tradeNo*.

User experience

To complete a payment in the mini program, users usually follow the steps below:

1. The user chooses goods in the mini program and create an order, then presses the **Pay** button in the mini program.
2. The mini program redirects the user to the wallet app and the wallet app displays the payment page.
3. The user confirms the order information, such as the payee and amount, then clicks the **Pay** button for payment.
4. After confirming the payment, the wallet app displays the payment result and redirects the user back to the payment result page in the mini program.

Procedures

To use the *my.tradePay* API to initiate a payment in the mini program, mini program developers must complete the following steps:

1. Confirm that the payment service provided by the wallet app supports the payment by *tradeNO* and obtain the integration guide from the wallet.
2. Integrate the payment service at the mini program server side.
3. Create a mini program in the wallet workspace on the Mini Program platform, or make sure that the mini program can be published in the wallet app.
4. Provide goods and payment service in the mini program.
5. Publish the mini program.

Payment process

The following figure illustrates the payment process:

Figure 1. Payment process

The payment process contains the following steps:

1. The user places an order in the mini program. (Step 1)
2. The mini program client sends the create order request to the mini program server by calling the my.request API. (Step 2)
3. The mini program server creates the order and sends the request to the wallet server via the server API provided by the wallet. (Step 3 & 4)
4. The wallet server creates the order and generates *tradeNo*, and returns *tradeNo* to the mini program server. (Step 5-7)
5. The mini program server returns *tradeNo* to the mini program client. (Step 8)
6. The mini program client initiates the payment request by calling the my.tradePay API with *tradeNo* to the wallet app. (Step 9)
7. The wallet app obtains the order information and renders the cashier page. (Step 10-12)
8. The user confirms the payment and the wallet processes the payment request. (Step 13-16)
9. The wallet app returns the payment result to the mini program client. At the same time, the wallet server returns the payment result to the mini program server. (Step 17 & 18)
10. The mini program client displays the payment result. (Step 19)

Note:

The payment flow is for reference and might differ depending on the API implementation of the wallet.

Sample code

Sample code for the my.tradePay API calling:

copy

```
my.tradePay({
    tradeNO: '20171115210011041053366779 2',
    success: function(res) {
        my.alert({
            content: JSON.stringify(res),
```

```
        });
    },
    fail: function(res) {
        my.alert({
            content: JSON.stringify(res),
        });
    },
});
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_openapi_pay-with-tradeno

---

User authorization {#user-authorization}

Last updated: 2022-07-07

Path: miniprogram_gcash

User authorization

2022-07-07 17:08

User authorization describes the process of obtaining a user's consent to access user information. It is based on the industry standard OAuth2.0 authorization mechanism. On the Mini Program Platform, developers need to get permission from users in the mini program before obtaining and using their information.

Terminology

| | | | --- | --- | | **Name** | **Description** | | Scope of authorization (scope) | A scope represents the scope of permissions that developers need to request user authorization. A scope contains at least one open API interface or JSAPI interface. One authorization can combine multiple scopes for combined authorization. | | Authorization code (auth_code) | Temporary user authorization credentials. After obtaining it, promptly exchange it for the access token mentioned below. | | Access token or authorization token (access_token or auth_token) | Long-term authorization credentials. It is used to call the site gateway to call the server-side authorization interface. Pay attention to the scope and validity of the authorization token. | | Refresh token (refresh_token) | Used to refresh and obtain a new access token after the access token expires. The refresh token also has a validity period. |

Scope list

| | | | --- | --- | | **Scope** | **Description** | | auth_base | Authorized to obtain the unique user ID. | | auth_user | Authorized to obtain the user information. |

Access guidelines

Access process

Take obtaining the user information as an example. The overall access process is illustrated as below:

1. The mini program calls the `getAuthCode` JSAPI to get the authorization code (`authCode`) from the wallet [1.1].
2. The mini program calls the merchant server API with `authCode` [2].
3. The merchant server calls the `applyToken` OpenAPI and the authorized platform server returns the access token [2.2].
4. The merchant server saves the access token and returns the authorization result to the mini program [2.4].

Note: To authorize other information, use a different scope for the scopes parameter when calling `getAuthCode`.

Obtain authCode

You can obtain user authorization by calling the my.getAuthCode JSAPI and fetch the `authCode` in the success callback. For example:

copy

```
my.getAuthCode({
  scopes: ['auth_user'],
  success: (res) => {
    my.alert({
      content: res.authCode,
    });
  },
  failed: (res) => {
      console.log(res.authErrorScopes)
  },
});
```

Obtain accessToken

- For merchants: Before obtaining an `accessToken`, you need to get an `authCode` from the wallet. Then you can call the `applyToken` OpenAPI in exchange for `accessToken`.
- For developers: Developers can exchange `accessToken` and `userId` with the obtained `authCode`.

Call the server OpenAPI

After obtaining the `accessToken`, developers can continue to use the access token to call other authorization interfaces. Pay attention to the permission scope and validity period of the token.

API List

| | | | --- | --- | | **JSAPI** | **Description** | | my.getAuthCode | Gets user's authorization code. | | **OpenAPI** | **Description** | | v1/authorizations/applyToken | Obtain the access token. |

FAQs

1. Why should developers use `my.getAuthCode` API?

All the reading and writing of user information on the Mini Program Platform can only be used after obtaining the user's consent. User authorization is based on the industry standard OAuth2.0 authorization mechanism. With this mechanism, developers can obtain user information on the Mini Program Platform.

2. Why is the user authorization API not allowed on the first screen of the mini program?

In order to create a better user experience on the mini program, user authorization guidance is not allowed on the first screen of the mini program. The guidance for user authorization should be given after the user fully understands the business content of the mini program. We recommend you add the mini program authorization into the business process.

3. Can `userId` be obtained through the user authorization API?

No, the `userId` needs to be obtained by calling the related API on the server side.

More information

Obtain basic user information

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_openapi_userauthorization

---

User authorization {#user-authorization}

Last updated: 2021-05-09

Path: miniprogram_gcash

User authorization

2021-05-09 18:43

Product Description

All the reading and writing of user information on the site open platform requires the user's permission before it can be used, user authorization is based on the international standard OAuth2.0 authorization mechanism. Based on this mechanism, developers can obtain site user information etc.

Terminology

| Terminology | Description | Remark | | scope | Scope of authorization | A scope represents the scope of permissions that developers need to request user authorization. A scope contains at least one openapi interface or JSAPI interface. One authorization can combine multiple scopes for combined authorization. | | auth_code | Authorization code | Temporary user authorization credentials, after obtaining it, please promptly exchange for the authorization token mentioned below. | | access_token/auth_token | Authorization token, or access token | Long-term authorization credentials are used to call the site gateway for server-side authorization interface calls. Need to pay attention to the scope and validity of authorization token. | | refresh_token | Refresh token | Used to refresh and obtain new authorization token after the authorization token expires, the refresh token also has a validity period. |

Related products

Obtain Basic Member Information

Scopes List

| Scopes | Description | | USER_ID | Authorized to obtain the unique user ID. | | USER_NICKNAME | Authorized to obtain the user nickname. | | USER_NAME | Authorized to obtain the user name. | | USER_LOGIN_ID | Authorized to obtain the user login ID. | | HASH_LOGIN_ID | Authorized to obtain the hash user login ID. | | USER_AVATAR | Authorized to obtain the user avatar. | | USER_GENDER | Authorized to obtain the user gender. | | USER_BIRTHDAY | Authorized to obtain the user birthday. | | USER_NATIONALITY | Authorized to obtain the user nationality. | | USER_CONTACTINFO | Authorized to obtain the user contact info. | | auth_base | Authorized to obtain the unique user ID. | | auth_user | Authorized to obtain user information. |

Access Guidelines

Access Process

Obtain user information as an example, the overall access process is as follows (if you need to authorize other information, you only need to use a different scope for the scopes parameter when calling getAuthCode).

App Obtains Authcode

The user authorization is obtained by calling the jsapi my.getAuthCode, and the authcode can be obtained in the success callback. The js code is as follows:

copy

```
my.getAuthCode({
  scopes: ['USER_ID'],
  success: (res) => {
    my.alert({
      content: res.authCode,
    });
  },
  failed: (res) => {
      console.log(res.authErrorScopes)
  },
});
```

Server Obtains Access Token

Merchant server can call `v1/authorizations/applyToken` interface in exchange for the access_token， developers can exchange access_token and userId with the obtained auth_code。auth_code as a ticket in exchange for access_token。

Call The Server Business API

After obtaining access_token, developers can continue to use the token to call other authorization interface. Please pay attention to the permission scope and timeliness of the token.

API List

| API | API Description | | my.getAuthCode | Obtain the authorization code. |

QA

Question: Why should developers must use my.getAuthCode API?

Answer： All the reading and writing of user information on the site open platform requires the user’s permission before it can be used， user authorization is based on the international standard OAuth2.0 authorization mechanism. Based on this mechanism, developers can obtain site user information etc.

Question: Why is it not allowed to use the user authorization API on the first screen of the Mini Program?

Answer： In order to create a better Mini Program user experience, guiding user authorization on the first screen of the Mini Program is not allowed. It is necessary to guide the user authorization after the user fully understands the business content of the Mini Program. It is recommended to put the Mini Program authorization in the business process.

Question: Can the userId be obtained through the user authorization API?

Answer： No, userId needs to be obtained by calling api on the server side.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_openapi_userauthorization

---

User information capability {#user-information-capability}

Last updated: 2022-07-03

Path: miniprogram_gcash

User information capability

2022-07-03 18:44

All the user information on the Mini Program Platform requires user's authorization. Based on the industry standard OAuth2.0 authorization mechanism, mini program developers can get user authorization to obtain user information.

Note: Developers must fully respect the privacy of users and properly use the user authorization. If the information is found to be used beyond the agreed scope or reasonable usage, the platform has the right to permanently withdraw the interface authority of mini program.

Prerequisites

- This capability is open to merchants who have become business partners of the wallet.
- Make sure that the integration and configuration have been completed and the mini program has been released.

Interaction process

Silence mode

The silence mode requires the user's consent on a native app to collect the required information. The interaction flow of the silence mode is illustrated as below:

1. The user opens the wallet app and is redirected to the merchant mini program.
2. The merchant mini program calls the `getAuthCode` JSAPI to request `authCode` from the wallet app.
3. The wallet app returns `authCode` to the merchant mini program, which sends `authCode` to the ISV or merchant server.
4. With the obtained `authCode` in step 3, the ISV or merchant server calls the `/{version}/oauths/applyToken` OpenAPI to request `accessToken` and `uid` from the wallet server.

Note: The `version` is the version of Open APIs, for example, `v1` or `v2`.

5. The wallet server returns `accessToken` and `uid` to the ISV or merchant server.

Notes:

- `authCode` is used to exchange for `accessToken`. Every time the user authorization is completed, `authCode` in the JSAPI response is different. `authCode` can only be used once and will automatically expire within one day.
- After the ISV or merchant obtains `accessToken` and `uid`:
- The ISV or merchant can use `accessToken` to call other OpenAPIs. For example, call the `inquiryUserInfoByAccessToken` OpenAPI to query the user information.
- The ISV or merchant can generate a **session** that maps to `accessToken` and `uid`, then set session expiration time and store the mapping. The session will be stored in the mini program framework.

User consent mode

The user consent mode is used to get public user information without further permission from wallets. The interaction flow of the user consent mode is illustrated as below:

Get user open info

When the merchant mini program intends to get some public information of users, such as name and avatar, use the user consent mode with the `getOpenUserInfo` JSAPI. The user needs to sign the agreement and clicks the **Accept** button. This function is used to display some personal data of the user in the mini program.

Get auth code

When the merchant mini program intends to get `authCode` for further usage, call the `getAuthCode` JSAPI by specifying the `scope` field.

API list

| | | | --- | --- | | **JSAPI** | **Description** | | my.getOpenUserInfo | Gets user basic information, such as avatar, nickname, etc. | | my.getAuthCode | Gets user's authentication code. | | **OpenAPI** | **Description** | | `/{version}/authorizations/{apiName}`
Note: The `version` is the version of Open APIs, for example, `v1` or `v2`. | For details, see the Open APIs for Merchants chapter. | | /{version}/users/inquiryUserInfo |

More information

Capabilities

JSAPIs

Open APIs

Developing Mini Program

Using Mini Program Platform

Features

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/capability-user-information

---

User permission requests {#user-permission-requests}

Last updated: 2021-05-10

Path: miniprogram_gcash

User permission requests

2021-05-10 04:11

To access a mini program user's device and user information, mini programs must obtain the user's permission. This document contains design guidelines for developers and designers to create a user-friendly journey for users to authorize mini programs to access the following types of information:

- Device information
- User information

A popup message will be displayed when the mini program requires a user's permission.

Device information

Device information includes the resources on user's devices, such as the current location, camera, microphone, photos, Bluetooth, and contacts. For device information, the following JSAPIs are available in the container for mini programs to call:

- my.getLocation
- my.chooseImage
- my.saveImage
- my.scan
- my.getOpenUserInfo

For more information, see **Developing Mini Programs** > **References** > **JSAPIs** > **Device** > **Settings**.

User information

User information includes a user's personal information, such as mobile phone number, basic information, wallet member information, points, coupons, and so on. For user information, the wallet app either implements or develops the corresponding APIs to be called for permission.

In some cases, user terms also need to be displayed and users need to agree with the terms for mini programs to access user information.

Design principles

Use the following principles as guidelines to design the user flow for requesting permissions:

- **Request permissions only when your mini program requires resources**

We do not suggest to request for permissions at the launch of the mini program, or request all permissions together at one time. However, if users must agree with user terms before using your mini program, you must display the terms when users enter your mini program for the first time.

- **Display clear choices for users to make**

Buttons must clearly distinguish the choices for users to make. The primary button is **Allow** and the secondary button is **Don't Allow**. It is not suggested to use **Cancel** as the secondary button because users will be confused about whether the permission is granted or not.

- **Explain the resources to be accessed specifically**

Provide texts to explain the resources to be accessed clearly. For example, display the user's mobile phone number to be accessed and the source of the number, or list the personal user information to be accessed.

- **Guide users to reverse the decision to grant permission**

If the user denies permission requests, you can explain the benefits of granting permissions in related scenarios and provide a link to where the user can reverse the decision.

Applicable scenarios

The following illustrate how these design principles can be applied.

Granting permissions is required

In this scenario, the user cannot enjoy the service provided by the mini program without granting permissions. To ensure that the user understands this, we suggest redirecting the user back to the previous page if the user clicks **Don't Allow**. When the user wants to use the same service, display the bottom sheet component requesting the permission again till the user grants the permission.

Granting permissions is not required

While granting permissions allows the user to have a smoother mini program experience, it is sometimes not necessary for the user to use certain mini program functions. For example, if a user grants the mini program permission to access the current location, the user would not have to manually input the address to use the function. When the user uses the same service in future, the bottom sheet component is displayed again to request the permission.

Options to ignore permission request or reverse decision

When granting the permission is not required, you can add the selection of **Do not ask again**. If the user checks the selection, the same permission request is not displayed again when the user does the same action within the mini program. However, if permissions are required for certain actions within the mini program, display the bottom sheet component to request the permission and guide the user to where the permission can be reversed.

Request permissions to link the wallet and merchant accounts

If a user has used a wallet app to sign up for a merchant mini program account, such as with an email address or phone number, the mini program can guide the user to link the mini program account with the wallet account.

Component and samples

Permissions requests are communicated to users via a bottom sheet component displayed at the bottom of the screen.

Bottom sheet

Bottom sheet is a UI component that slides from the bottom of the screen when the user makes a specific action. On the bottom sheet, the user can start a new task, make a choice, or confirm the to-do action. The bottom sheet component is significantly less disruptive than the popup modal, which appears in the middle of the screen.

The bottom sheet used to request for permissions consists of three parts:

- Mini program name: A right arrow is placed on the right of the mini program name to indicate that the mini program name is a link. The mini program name is linked to the **About Us** page.
- Requested resources: All requested resources and icons are listed clearly.
- Buttons: Display two opposite choices for users to make, such as **Allow** and **Don't Allow**.

As the mini program must request permissions each time the user uses the mini program, we suggest adding the **Remember my choice** selection so that users will not be prompted again.

Samples

Permission requests for resources on the user's mobile device

Permission requests for the user device include the access to contacts, photos, current location, microphone, Bluetooth, and so on. See the following UI samples for details.

Permission requests for the user information

User information includes the user's account, avatar, phone number, and so on. When accessing the user information, comply with the following rules:

- Explain the source of the information if required. For example, the phone number is retrieved from the phone's address book.
- Provide a button to allow users to change the information if required. For example, when accessing the user account or phone number, provide an option for the user to use another account or phone number.
- Provide a link to the terms for users as a reference.

See the following UI samples for details.

Use case

Vodapay, a digital wallet, cooperates with more merchants in the form of third-party mini programs. When users use the merchant mini program in the Vodapay wallet app for the first time, the merchant mini program requests all required permissions so that users can enjoy services provided by the mini program. At the same time, users must sign the terms provided by the merchant before using the mini program.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/design/user-permission-request

---

VTabs {#vtabs}

Last updated: 2022-07-03

Path: miniprogram_gcash

VTabs

2022-07-03 18:44

Tabs allow the user to switch between different views.

Vtabs

| | | | | | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Required** | | activeTab | Index of the currently active tab. | Number | Yes | | tabs | tab data, including the tab `title`, unique list anchor value, as well as the badge type `badgeType`, which includes `dot` and `text`, and is not displayed if the `badgeType` is not set. Badge text `badgeText` takes effect when the `badgeType` is `text`. | Array | Yes | | swipeable | An indicator of whether the tab can be swiped or not. | Boolean | Yes | | tabBarActiveBgColor | tabBar background color in active status. | String | No | | tabBarInactiveBgColor | tabBar background color in inactive status. | String | No | | tabBarActiveTextColor | Active Tab text color of the tabBar. | String | No | | tabBarInactiveTextColor | Inactive Tab text color of the tabBar. | String | No | | tabBarlineColor | tabBar sideline color. | String | No | | onTabClick | Callback when the tab is clicked. | (index: Number) => void | No | | onChange | Trigger on vtab-content change. | (index: Number) => void | No |

Vtab-content

View content

| | | | | | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Required** | | anchor | Unique anchor value of list. | String | Yes |

Example

copy

```
{
  "defaultTitle": "AntUI Component Library",
  "usingComponents": {
    "vtabs": "mini-antui/es/vtabs/index",
    "vtab-content": "mini-antui/es/vtabs/vtab-content/index"
  }
}
```

copy

```
<view>
  <vtabs
    tabs="{{tabs}}"
    onTabClick="handleChange"
    onChange="onChange"
    activeTab="{{activeTab}}"
  >
    <block a:for="{{tabs}}">
      <vtab-content anchor="{{item.anchor}}">
        <view style="border: 1px solid #eee; height: 800px; box-
sizing: border-box">
          <text>content of {{item.title}}</text>
        </view>
      </vtab-content>
    </block>
  </vtabs>
</view>
```

copy

```
Page({
  data: {
    activeTab: 2,
    tabs: [\
      { title: 'Option two', anchor: 'a', badgeType: 'dot' },\
      { title: 'Option', anchor: 'b', badgeType: 'text', badgeText:
'New' },\
      { title: 'Option three', anchor: 'c' },\
      { title: 'Option four', anchor: 'd' },\
      { title: 'Option five', anchor: 'e' },\
      { title: 'Option six', anchor: 'f' },\
    ],
  },
  handleChange(index) {
    this.setData({
      activeTab: index,
    });
  },
```

```
  onChange(index) {
    console.log('onChange', index);
    this.setData({
      activeTab: index,
    });
  },
});
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_layout-navigation_vtabs

---

Verify Code {#verify-code}

Last updated: 2022-07-03

Path: miniprogram_gcash

Verify Code

2022-07-03 18:44

You can use the verify-code component to display the input box of the verification code.

Note:

The verify-code is a controlled component. The component value needs to be obtained by the *onInput* event.

Sample code

See the sample codes in different languages:

.json

copy

```
{
  "defaultTitle": "Verify-code",
  "usingComponents": {
    "verify-code": "mini-ali-ui/es/verify-code/index"
  }
}
```

.axml

copy

```
<view>
  <view style="margin-top: 10px;" />
  <view style="padding: 0 10px;">verify code box</view>
  <view style="margin-top: 10px;" />
  <verify-code
    onInput="onInput"
    value="{{verifyCode}}"
    onClear="onClear"
    last="{{true}}"
    countDown="{{10}}"
    initActive="{{false}}"
    onSend="onSend"></verify-code>
</view>
```

.js

copy

```
Page({
  data: {
    verifyCode: '',
  },
  onSend() {
    my.alert({
      title: 'verify code sent',
    });
  },
  onInput(e) {
    this.setData({
      verifyCode: e.detail.value,
    });
  },
});
```

Parameters

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | className | String | Customized class. | | label | String | Customized label text. The default value is `Verification Code`. | | labelCls | String | Customized class for the label. | | inputCls | String | Customized class for the input box. | | last | Boolean | An indicator of whether the input box is the last one. The default value is `false`. | | value | String | Input box value. | | name | String | Component name, which is used to obtain data by submitting the form. | | placeholder | String | Placeholder. | | placeholderStyle | String | Style of the placeholder. | | placeholderClass | String | Style class of the placeholder. | | disabled | Boolean | An indicator of whether to disable the function of clearing the verification code. The default value is `false`. | | maxlength | Number | Maximum length of the verification code. The default value is `140`. | | focus | Boolean | An indicator of whether to get focus. The default value is `false`. | | clear | Boolean | An indicator of whether to clear the input. The default value is `true`, and takes effect only when the value of *disabled* is `false`. | | onInput | (e: Object) => void | The event that is triggered when users tap the keyboard. | | onConfirm |

(e: Object) => void | The event that is triggered when users tap the **Done** button on the keyboard. | | onFocus | (e: Object) => void | The event that is triggered when an element gets the focus. | | onBlur | (e: Object) => void | The event that is triggered when an element loses the focus. | | onClear | () => void | The event that is triggered when users tap the **Clear** button. | | txtSend | String | Text on the send verification code button. The default value is **Send**. | | txtSendAgain | String | Text on the resend verification code button. The default value is **Resend**. | | txtCountDown | String | Counting down text before resending the verification code, which does not include the time. The default value is **Resend later**. | | initActive | Boolean | An indicator of whether to trigger the send button actively. The default value is `false`. When the value is `true`, count down the resending time automatically after the component is initially loaded. You can set the prompt information according to your requirements. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_form_verify-code

---

Video Tutorial: Getting Started with Mini Programs {#video-tutorial:-getting-started-with-mini-programs}

Last updated: 2022-07-07

Path: miniprogram_gcash

Video Tutorial: Getting Started with Mini Programs

2022-07-07 17:08

The Mini Program Platform provides comprehensive functionalities for you to develop and manage mini programs. To have a quick start for the mini program platform, you can watch the video tutorials.

Manage mini programs

For a quick overview of how to manage mini programs, you can watch a video here:

For more information about the introduction to the Mini Program Manage functionality, see Manage Mini programs.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/getting-started/videos

---

Video Tutorial: Getting Started with Mini Programs {#video-tutorial:-getting-started-with-mini-programs}

Last updated: 2022-07-07

Path: miniprogram_gcash

Video Tutorial: Getting Started with Mini Programs

2022-07-07 17:08

The Mini Program Platform provides comprehensive functionalities for you to develop and manage mini programs. To have a quick start for the mini program platform, you can watch the video tutorials.

Manage mini programs

For a quick overview of how to manage mini programs, you can watch a video here:

For more information about the introduction to the Mini Program Manage functionality, see Manage Mini programs.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/getting-started/videos?pageVersion=2

Wallet onboarding {#wallet-onboarding}

Last updated: 2022-07-07

Path: miniprogram_gcash

Wallet onboarding

2022-07-07 17:08

Onboarding checklist

For wallets and other native apps, make sure the following steps are finished before you start using the platform.

As illustrated above, the wallet onboarding process consists of the following procedures:

1. Contact us to create a Mini Program workspace for your app.

In order for us to create a workspace, you're required to provide the following:

1. Your valid business license and point of contact
2. Your APK signature, bundle ID, and package name
3. Customize your workspace.

Once your Mini Program workspace creation request is approved, you will receive a dedicated URL to access your workspace in the Mini Program Platform.You then make the following configuration settings:

1. Replace the default URL to your business domain by configuring a reverse proxy server.
2. Customize the UI by replacing the default logo/favicon with your preferred logo.
3. Customize the email service.
4. Integrate the SDK (Android and iOS).

Download the configuration guide from your Platform and follow the steps in this guide to initialize the SDK. For more information, see *AC SDK Integration Guide*.

4. Implement the standard JSAPIs.

While the SDK provides you with the core Mini Program functionality, you can get enhanced features with a customized implementation of standard JSAPIs. For instance, when a Mini Program invokes the payment function, you are recommended to launch the existing standard payment flow instead of a completely new one. For more information, see JSAPIs.

5. Implement the standard Open APIs.

In addition to the client-side JSAPIs, you also need to implement the server-side APIs, in order to enable the capabilities, such as OAuth and Payment. For more information, see Open APIs.

Now you can start using the Mini Program Platform.

Next steps

Merchant onboarding

Contact us

If you're interested in using Mini Programs and the Mini Program Platform, please send us an email.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/getting-started/wallet-onboarding

---

What's New {#what's-new}

Last updated: 2021-05-10

Path: miniprogram_gcash

What's New

2021-05-10 04:19

What's New provides you with new, enhanced or deleted product features and system capabilities.

2021.4.27

Enhanced features

In the Quality page, you can see statistics for HTTP requests, JSAPI calls, JS errors, and details of abnormal URLs.

2021.4.14

Enhanced features

In the list page of Manage Mini Program, you can view the past performance, real-time analysis, and quality of each mini program and navigate to these pages.

In the detail page of a mini program, you can see the expiration time of the QR code for testing.

In the Real-Time Analysis page, you can see the success rate and error details of JSAPI calls.

Deleted features

The **Launch** feature is no longer available from the navigation menu.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/history/release-note

---

Workflow procedures {#workflow-procedures}

Last updated: 2024-12-24

Path: miniprogram_gcash

Workflow procedures

2024-12-24 21:43

This topic introduces the whole process that developers must go through to release a mini program.

Apply for an Account

Before developing a Mini Program, you need to create a developer account, which allows you to create, develop or publish your Mini Program.

Fill in the information, then you can finish the account creation process.

Ask the workspace admin to the role for you: Workspace Developer， Workspace Admin or Workspace Reviewer. Note that developer can develop the Mini Program, and only the Mini Program admin can create and publish the Mini Program.

Logging in

After completing the registration, log in directly at the login entry on the homepage of the Mini Program Platform.

Create a Tenant Workspace

Register a Workspace in Mini Program Development Platform (MPDP)

Step 1 Provide the information of your organization

Please provide the following information and send us email.

- Workspace Name
- Scope of Business (Please find the attachment📎MCC_20190701.xlsx, and send us the code)
- Business Address
- Business Representative Name & Contact Email (You can provide more than one. He/She will become the primary workspace admin of the platform)

Step 2 Be patient

It will take us 5 business days to generate a workspace and customized Mini Program Studio. Please be patient and reach us if you have any question.

Step 3 Your Tenant Workspace is generated

We will send you back a confirmation email with workspace account information once the workspace is generated.

Invite members and set roles

Give members controlled access to your Workspace

You can invite other members to access your workspace. To protect your sensitive information or restrict the actions they can perform, user roles limit their access. Each member must be assigned a role(In tenant space, you can set the user to be Workspace admin， Workspace reviewer， Workspace Developer) when they are added.

Team members and user roles are managed in your workspace "Members" settings page. You can add new members individually, or invite multiple users at the same time.

The "Members: settings page only allowed for workspace admin to view and manage.

Workspace

Workspace - General

Path: Workspace Manage

| | | | --- | --- | | **Role** | **Authorization** | | Workspace Admin | - Able to view the workspace informations. |

When you first created a Workspace by providing your company’s info., our team would generate a workspace with listing all the info. you have provided on the Workspace - general tab

| | | | --- | --- | | **Field** | **Description** | | Workspace ID | Generated by Platform. | | Workspace Name | Rule of naming:
- You can’t change the name after workspace is generated
- Character combination A-Z, a-z, 0-9
- No space

- Will be used as URL | | Status | | | Business Address | | | Scope of Business | Select from the attachment.

📎 MCC_20190701.xlsx | | Business Representative | The business representative will be the Workspace Admin as well. | | Representative email | |

-You can contact us to request for information change.

App Manage

Path: App Manage - App detail

| | | | --- | --- | | **Role** | **Authorization** | | Workspace Admin | - Able to add/view App infornation. |

| | | | --- | --- | | **Field** | **Description** | | Mini Program Accessing Key | Mini Program Accessing Key:
An unique key allowing container SDKs to access Mini Program data through SaaS. | | Mini Program Upload Key | Mini Program Upload Key:
- An unique key allowing Mini Program Bundles to build and upload to the Console.
- Container verification.
When you are ready to publish your mini program, you need to sign yourCopy mini program and upload it to a workspace portal. |

Members

For more information, see Workspace member roles.

Path: Workspace - Members

| | | | --- | --- | | **Role** | **Authorization** | | Workspace Admin | - Able to add/view the member(s).
- Able to set member(s) as other roles. |

| | | | --- | --- | | **Field** | **Description** | | Member's login email | User's registered email. | | Username | | | Role | Workspace roles:
Workspace Developer,
Workspace Admin,
Workspace Reviewer.
- When users first join the workspace, he/she would assign as Developer (Assigned to Mini Program: unassigned), Workspace Admin could set him/her as other roles, such as Workspace Admin;
Mini program Admin(Assigned to Mini Program: unassigned): Able to create a new mini program. | | Status | Member Status:
1. Active; corresponding action: block.
2. Inactive; corresponding action: unblock.
- Even the user got blocked, he/she could still log into the console. But the user would not able to access to the workspace and mini program on the console and Mini Program Studio. | | Start Date | When user first joined the workspace. | | Action | - Set as

Workspace Admin/Workspace Reviewer/Workspace Developer.
- Block,

Confirm notification: If the member has been blocked, he/she will not be able to view or operate all mini programs he/she has been joined.
Are you sure you want to block the member?
- Unblock, Confirm notification: If you unblock the member, he/she will be able to view or operate all mini programs he/she has been joined.

Are you sure you want to unblock the member? |

Approvals

Path: Approvals

| | | | --- | --- | | **Role** | **Authorization** | | Workspace Admin | - Able to view the approval list.
- Able to approve/reject application(s). | | Workspace Reviewer | - Able to view the approval list.
- Able to approve/reject application(s). |

| | | | | | --- | --- | --- | --- | | **Reviewing Category** | **English Title** | **Initiator Role** | **Approver Role** | | Mini Program Publishing | Apply for publishing review | Mini Program Admin | Workspace Admin | | Mini Program Removal | Apply for removal | Mini Program Admin | Workspace Admin |

| | | | --- | --- | | **Field** | **Description** | | Title | Title:
Apply for publishing review.
Apply for removal.
- Click to direct to the application page. | | Category | Reviewing Category:
Member requests,
Launch/Publishing,
Removal,
Feature activation. | | Mini Program name | Mini Program name. | | Version | Mini Program version. | | Applicant | Initiator role: Mini Program Admin. | | Created | Application created time. | | Finish time | Application reviewed time. | | Status | Review Status

In Review:
- After applicant submitted the application and before the reviewing.

Approved:
- After Approver reviewed.

Rejected:
- After Approver reviewed.

Withdrawn:
- Before the approver reviewed, the applicant could withdraw the application. |

More information

Overview

Member Role and Authorization

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/platform/workflow-procedures

---

Workspace member roles {#workspace-member-roles}

Last updated: 2022-07-07

Path: miniprogram_gcash

Workspace member roles

2022-07-07 17:08

This topic provides you with an overview of the member roles and the main duties of roles in the tenant workspace and developer workspace. Before you dive into the following sections, check the definitions below:

- **Tenant workspace** is where native apps manage their mini programs.
- **Developer workspace** is where merchants manage their mini programs.

Overview

A role is a set of defined access permissions. It can be used to grant authorizations to members who join a workspace on Mini Program Platform. Based on the assigned role, a member can perform specific operations in the workspace.

Default roles

There are total seven roles on Mini Program Platform:

For more details about each role, see the following sections.

Tenant workspace

The tenant workspace has four roles that can be assigned to members:

- **Workspace Admin** is the super administrator in charge of workspaces and the whole life cycle of mini programs of the native app, who is usually the first one to apply for joining the platform. Workspace admin can invite members to join the platform and assign roles to members. In addition, the role can also see mini programs of merchants.
- **Workspace Reviewer** is responsible for approvals from the native app and merchants, including requests related to mini programs and marketing. The role needs to review whether requests are in compliance with the local regulatory requirements.

- **Workspace Developer** is responsible for developing and releasing mini programs. The role can also add features for mini programs and JSAPIs to Mini Program Platform.
- **Workspace Operator** is responsible for managing the operation and marketing of mini programs.

Wallet members can access different operations based on assigned roles. For more information, see Member role authorization in tenant workspace.

Developer workspace

The developer workspace has three roles that can be assigned to members:

- **Developer Admin** is the super administrator in charge of the whole life cycle of mini programs of a merchant, who is usually the first one to apply for joining the platform as a merchant. Developer admin can invite members to join the platform and assign roles to members
- **Developer** is responsible for developing and releasing mini programs. The role can also monitor the quality and performance of mini programs.
- **Operator** is responsible for managing the operation and marketing of mini programs.

Merchant members can access different operations based on assigned roles. For more information, see Member role authorization in developer workspace.

More information

Workflow procedures

How to manage mini programs

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/platform/member-role

---

api/device/battery/getbatteryinfo {#api/device/battery/getbatteryinfo}

Path: miniprogram_gcash

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_device_battery_getbatteryinfo

---

api/event/onappshow {#api/event/onappshow}

Path: miniprogram_gcash

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_event_onappshow

---

api/ui/feedback/confirm {#api/ui/feedback/confirm}

Path: miniprogram_gcash

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_ui_feedback_confirm

---

api/ui/navigationbar/hidenavigationbarloading {#api/ui/navigationbar/hidenavigationbarloading}

Path: miniprogram_gcash

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_ui_navigationbar_hidenavigationbarloading

---

app-container {#app-container}

Path: miniprogram_gcash

404 Not Found

Sorry, the page you visited does not exist.

traceId: 21b1c25c17474850311331803ef3bb

Go Back

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/app-container

---

button {#button}

Last updated: 2022-07-03

Path: miniprogram_gcash

button

2022-07-03 18:44

Button.

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Default** | **Description** | | size | String | default | Effective value default, mini. | | type | String | default | Button style type, effective value primary, default, warn. | | plain | Boolean | false | Hollow or not. | | disabled | Boolean | false | Disable or not. | | loading | Boolean | false | Button text preceded with loading icon or not. | | hover-class | String | button-hover | Button pressed style class hover-class="none" indicates no pressed effect. | | hover-start-time | Number | 20 | Pressed status shown in a period after being pressed, in milliseconds. | | hover-stay-time | Number | 70 | Pressed status retention time after release, in milliseconds. | | form-type | String | | Effective value submit and reset, used for component, clicking triggers submit/reset event respectively. | | onTap | EventHandle | | Click. | | open-type | String | | Open ability. | | scope | String | | Valid when open-type is getAuthorize. |

The Valid Value of open-type

| | | | --- | --- | | **Value** | **Description** | | getAuthorize | Support for Mini Program authorization. |

The Valid Value of Scope

When `open-type` is `getAuthorize` , we can set scope to the following value:

| | | | --- | --- | | **Value** | **Description** | | userInfo | et user basic information. | | phoneNumber | Get user's phone number. |

Screenshot

Sample Code

copy

```
<view class="page">
  <view class="section">
    <view class="title">Type</view>
    <button type="default">default</button>
    <button type="primary">primary</button>
    <button type="warn">warn</button>
  </view>
  <view class="section" style="background:#ddd;">
    <view class="title">Misc</view>
    <button type="default" plain>plain</button>
    <button type="default" disabled>disabled</button>
    <button type="default" loading={{true}}>loading</button>
    <button type="default" hover-class="red">hover-red</button>
  </view>
  <view class="section">
    <view class="title">Size</view>
    <button type="default" size="mini">mini</button>
```

```
  </view>
  <view class="section">
    <view class="title">Type</view>
    <form onSubmit="onSubmit" onReset="onReset">
      <button form-type="submit">submit</button>
      <button form-type="reset">reset</button>
    </form>
  </view>
</view>
```

九色鹿

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/component_form-component_button

---

button {#button}

Last updated: 2021-05-09

Path: miniprogram_gcash

button

2021-05-09 18:43

Button.

Scan QR code to try:

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Default** | **Description** | | size | String | default | Effective value default, mini. | | type | String | default | Button style type, effective value primary, default, warn. | | plain | Boolean | false | Hollow or not. | | disabled | Boolean | false | Disable or not. | | loading | Boolean | false | Button text preceded with loading icon or not. | | hover-class | String | button-hover | Button pressed style class hover-class="none" indicates no pressed effect. | | hover-start-time | Number | 20 | Pressed status shown in a period after being pressed, in milliseconds. | | hover-stay-time | Number | 70 | Pressed status retention time after release, in milliseconds. | | form-type | String | | Effective value submit and reset, used for component, clicking triggers submit/reset event respectively. | | onTap | EventHandle | | Click. | | open-type | String | | Open ability. | | scope | String | | Valid when open-type is getAuthorize. |

The Valid Value of open-type

| | | | --- | --- | | **Value** | **Description** | | getAuthorize | Support for Mini Program authorization. |

The Valid Value of Scope

When `open-type` is `getAuthorize` , we can set scope to the following value：

| | | | --- | --- | | **Value** | **Description** | | userInfo | et user basic information. | | phoneNumber | Get user's phone number. |

Screenshot

Sample Code

copy

```
<view class="page">
  <view class="section">
    <view class="title">Type</view>
    <button type="default">default</button>
    <button type="primary">primary</button>
    <button type="warn">warn</button>
  </view>
  <view class="section" style="background:#ddd;">
    <view class="title">Misc</view>
    <button type="default" plain>plain</button>
    <button type="default" disabled>disabled</button>
    <button type="default" loading={{true}}>loading</button>
    <button type="default" hover-class="red">hover-red</button>
  </view>
  <view class="section">
    <view class="title">Size</view>
    <button type="default" size="mini">mini</button>
  </view>
  <view class="section">
    <view class="title">Type</view>
    <form onSubmit="onSubmit" onReset="onReset">
      <button form-type="submit">submit</button>
      <button form-type="reset">reset</button>
    </form>
  </view>
</view>
```

九色鹿

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/component_form-component_button

---

canvas {#canvas}

Last updated: 2022-07-03

Path: miniprogram_gcash

canvas

2022-07-03 18:44

Canvas.

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Default** | **Description** | | id | String | | Unique component identifier. | | style | String | | | | class | String | | | | width | String | canvas width attribute | | | height | String | canvas height attribute | | | disable-scroll | Boolean | false | Forbid screen scroll and pull-to-refresh. | | onTap | EventHandle | | Click. | | onTouchStart | EventHandle | | Touch action start. | | onTouchMove | EventHandle | | Move after touch. | | onTouchEnd | EventHandle | | Touch action end. | | onTouchCancel | EventHandle | | Touch action interrupted, such as incoming call reminder, pop-up. | | onLongTap | EventHandle | | Trigger on long-press for 500ms, thereafter move action does not trigger screen scroll. |

Note:

- The canvas tab has default width 300px and height 225px
- On the same page, the id cannot be repeated.
- For finer displaying in higher dpr, use the attribute settings to zoom in and use the style to zoom out the canvas. for example:

copy

```
<!-- getSystemInfoSync().pixelRatio === 2 -->
<canvas width="200" height="200" style="width:100px;height:100px;"/>
```

Screenshot

Sample Code

copy

```
<canvas
  id="canvas"
  class="canvas"
  onTouchStart="log"
  onTouchMove="log"
  onTouchEnd="log"
/>
```

copy

```
Page({
  onReady() {
```

```
    this.point = {
      x: Math.random() * 295,
      y: Math.random() * 295,
      dx: Math.random() * 5,
      dy: Math.random() * 5,
      r: Math.round(Math.random() * 255 | 0),
      g: Math.round(Math.random() * 255 | 0),
          b: Math.round(Math.random() * 255 | 0),
    };
    this.interval = setInterval(this.draw.bind(this), 17);
  },
  draw() {
    var ctx = my.createCanvasContext('canvas');
    ctx.setFillStyle('#FFF');
    ctx.fillRect(0, 0, 305, 305);
    ctx.beginPath();
    ctx.arc(this.point.x, this.point.y, 10, 0, 2 * Math.PI);
    ctx.setFillStyle("rgb(" + this.point.r + ", " + this.point.g + ",
" + this.point.b + ")");
    ctx.fill();
    ctx.draw();
    this.point.x += this.point.dx;
    this.point.y += this.point.dy;
    if (this.point.x <= 5 || this.point.x >= 295) {
      this.point.dx = -this.point.dx;
      this.point.r = Math.round(Math.random() * 255 | 0);
      this.point.g = Math.round(Math.random() * 255 | 0);
      this.point.b = Math.round(Math.random() * 255 | 0);
    }
    if (this.point.y <= 5 || this.point.y >= 295) {
      this.point.dy = -this.point.dy;
      this.point.r = Math.round(Math.random() * 255 | 0);
      this.point.g = Math.round(Math.random() * 255 | 0);
      this.point.b = Math.round(Math.random() * 255 | 0);
    }
  },
  drawBall() {
  },
  log(e) {
    if (e.touches && e.touches[0]) {
      console.log(e.type, e.touches[0].x, e.touches[0].y);
    } else {
      console.log(e.type);
    }
  },
  onUnload() {
    clearInterval(this.interval)
  }
})
```

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/component_canvas_canvas

---

canvas {#canvas}

Last updated: 2021-05-09

Path: miniprogram_gcash

canvas

2021-05-09 18:43

Canvas.

Scan QR code to try:

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Default** | **Description** | | id | String | | Unique component identifier. | | style | String | | | | class | String | | | | width | String | canvas width attribute | | | height | String | canvas height attribute | | | disable-scroll | Boolean | false | Forbid screen scroll and pull-to-refresh. | | onTap | EventHandle | | Click. | | onTouchStart | EventHandle | | Touch action start. | | onTouchMove | EventHandle | | Move after touch. | | onTouchEnd | EventHandle | | Touch action end. | | onTouchCancel | EventHandle | | Touch action interrupted, such as incoming call reminder, pop-up. | | onLongTap | EventHandle | | Trigger on long-press for 500ms, thereafter move action does not trigger screen scroll. |

Note:

- The canvas tab has default width 300px and height 225px
- On the same page, the id cannot be repeated.
- For finer displaying in higher dpr, use the attribute settings to zoom in and use the style to zoom out the canvas. for example:

copy

```
<!-- getSystemInfoSync().pixelRatio === 2 -->
<canvas width="200" height="200" style="width:100px;height:100px;"/>
```

Screenshot

Sample Code

copy

```
<canvas
  id="canvas"
  class="canvas"
  onTouchStart="log"
  onTouchMove="log"
  onTouchEnd="log"
/>
```

copy

```
Page({
  onReady() {
    this.point = {
      x: Math.random() * 295,
      y: Math.random() * 295,
      dx: Math.random() * 5,
      dy: Math.random() * 5,
      r: Math.round(Math.random() * 255 | 0),
      g: Math.round(Math.random() * 255 | 0),
          b: Math.round(Math.random() * 255 | 0),
    };
    this.interval = setInterval(this.draw.bind(this), 17);
  },
  draw() {
    var ctx = my.createCanvasContext('canvas');
    ctx.setFillStyle('#FFF');
    ctx.fillRect(0, 0, 305, 305);
    ctx.beginPath();
    ctx.arc(this.point.x, this.point.y, 10, 0, 2 * Math.PI);
    ctx.setFillStyle("rgb(" + this.point.r + ", " + this.point.g + ",
" + this.point.b + ")");
    ctx.fill();
    ctx.draw();
    this.point.x += this.point.dx;
    this.point.y += this.point.dy;
    if (this.point.x <= 5 || this.point.x >= 295) {
      this.point.dx = -this.point.dx;
      this.point.r = Math.round(Math.random() * 255 | 0);
      this.point.g = Math.round(Math.random() * 255 | 0);
      this.point.b = Math.round(Math.random() * 255 | 0);
    }
    if (this.point.y <= 5 || this.point.y >= 295) {
      this.point.dy = -this.point.dy;
      this.point.r = Math.round(Math.random() * 255 | 0);
      this.point.g = Math.round(Math.random() * 255 | 0);
      this.point.b = Math.round(Math.random() * 255 | 0);
    }
  },
  drawBall() {
  },
  log(e) {
    if (e.touches && e.touches[0]) {
```

```
      console.log(e.type, e.touches[0].x, e.touches[0].y);
    } else {
      console.log(e.type);
    }
  },
  onUnload() {
    clearInterval(this.interval)
  }
})
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/component_canvas_canvas

---

checkbox {#checkbox}

Last updated: 2021-05-09

Path: miniprogram_gcash

checkbox

2021-05-09 18:43

Scan QR code to try:

checkbox-group

Multiple selector group

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | name | String | Component name, used to form submission to get data. | | onChange | EventHandle | Trigger on change of checked item, detail = {value: Value of the checked checkbox item}. |

checkbox

Multiple choice

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Default** | **Description** | | value | String | | Component value, value carried in change event when checked. | | checked | Boolean | false | Checked or not, used to set checked by default. | | disabled | Boolean | false | Disable or not. | | onChange | EventHandle | | Trigger on change of component, detail = {value: Is the checkbox checked or not}. | | color | Color | | Checkbox color. |

Screenshot

Sample Code

copy

```
// acss
.checkbox {
  display: block;
  margin-bottom: 20rpx;
}
.checkbox-text {
  font-size:34rpx;
  line-height: 1.2;
}
```

copy

```
<checkbox-group onChange="onChange">
  <label class="checkbox" a:for="{{items}}">
    <checkbox value="{{item.name}}" checked="{{item.checked}}"
disabled="{{item.disabled}}" />
    <text class="checkbox-text">{{item.value}}</text>
  </label>
</checkbox-group>
```

copy

```
Page({
  data: {
    items: [\
      {name: 'angular', value: 'AngularJS'},\
      {name: 'react', value: 'React', checked: true},\
      {name: 'polymer', value: 'Polymer'},\
      {name: 'vue', value: 'Vue.js'},\
      {name: 'ember', value: 'Ember.js'},\
      {name: 'backbone', value: 'Backbone.js', disabled: true},\
    ],
  },
  onChange(e) {
    my.alert({
      title: `You are selecting the framework ${e.detail.value}`,
    });
  },
});
```

九色鹿

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/component_form-component_checkbox

---

checkbox {#checkbox}

Last updated: 2022-07-03

Path: miniprogram_gcash

checkbox

2022-07-03 18:44

checkbox-group

Multiple selector group

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | name | String | Component name, used to form submission to get data. | | onChange | EventHandle | Trigger on change of checked item, detail = {value: Value of the checked checkbox item}. |

checkbox

Multiple choice

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Default** | **Description** | | value | String | | Component value, value carried in change event when checked. | | checked | Boolean | false | Checked or not, used to set checked by default. | | disabled | Boolean | false | Disable or not. | | onChange | EventHandle | | Trigger on change of component, detail = {value: Is the checkbox checked or not}. | | color | Color | | Checkbox color. |

Screenshot

Sample Code

copy

```
// acss
.checkbox {
  display: block;
  margin-bottom: 20rpx;
}
.checkbox-text {
  font-size:34rpx;
  line-height: 1.2;
}
```

copy

```
<checkbox-group onChange="onChange">
  <label class="checkbox" a:for="{{items}}">
    <checkbox value="{{item.name}}" checked="{{item.checked}}"
disabled="{{item.disabled}}" />
    <text class="checkbox-text">{{item.value}}</text>
  </label>
</checkbox-group>
```

copy

```
Page({
  data: {
    items: [\
      {name: 'angular', value: 'AngularJS'},\
      {name: 'react', value: 'React', checked: true},\
      {name: 'polymer', value: 'Polymer'},\
      {name: 'vue', value: 'Vue.js'},\
      {name: 'ember', value: 'Ember.js'},\
      {name: 'backbone', value: 'Backbone.js', disabled: true},\
    ],
  },
  onChange(e) {
    my.alert({
      title: `You are selecting the framework ${e.detail.value}`,
    });
  },
});
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/component_form-component_checkbox

---

component/form-component/label {#component/form-component/label}

Path: miniprogram_gcash

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/component_form-component_label

---

develop-miniprogram {#develop-miniprogram}

Path: miniprogram_gcash

404 Not Found

Sorry, the page you visited does not exist.

traceId: 218402da17474851391032157e7fc3

Go Back

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/develop-miniprogram

---

error/codes {#error/codes}

Path: miniprogram_gcash

404 Not Found

Sorry, the page you visited does not exist.

traceId: 21b1c24e17474850757413388e8609

Go Back

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/error_codes

---

extended-component/others/loading {#extended-component/others/loading}

Path: miniprogram_gcash

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_others_loading

---

form {#form}

Last updated: 2022-07-03

Path: miniprogram_gcash

form

2022-07-03 18:44

Form, used to submit the user entry textarea, switch, input, checkbox-group, slider, radio-group, picker and other components in the component.

Clicking the button component with “form” form and “form-type” as “submit” causes submission of the “value” value in the form component. It is required to add “name” in the form component as the key.

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | onSubmit | EventHandle | Carrying data in form triggers submit event, event.detail = {value : {'name': 'dao14'}, buttonTarget: {'dataset': 'buttonDataset'} } . | | onReset | EventHandle | Trigger reset event upon form reset. |

Screenshot

Sample Code

copy

```
<form onSubmit="formSubmit" onReset="formReset">
  <view class="section section_gap">
    <view class="section__title">switch</view>
    <switch name="switch"/>
  </view>
  <view class="section section_gap">
    <view class="section__title">slider</view>
    <slider name="slider" show-value ></slider>
  </view>
  <view class="section">
    <view class="section__title">input</view>
    <input name="input" placeholder="please input here" />
  </view>
  <view class="section section_gap">
    <view class="section__title">radio</view>
    <radio-group name="radio-group">
      <label><radio value="radio1"/>radio1</label>
      <label><radio value="radio2"/>radio2</label>
    </radio-group>
  </view>
  <view class="section section_gap">
    <view class="section__title">checkbox</view>
    <checkbox-group name="checkbox">
      <label><checkbox value="checkbox1"/>checkbox1</label>
      <label><checkbox value="checkbox2"/>checkbox2</label>
    </checkbox-group>
  </view>
  <view class="btn-area">
    <button formType="submit">Submit</button>
    <button formType="reset">Reset</button>
  </view>
</form>
```

copy

```
Page({
  formSubmit: function(e) {
    console.log('form has a submit event, carrying data ',
e.detail.value)
  },
  formReset: function() {
    console.log('form has a reset event')
  }
})
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/component_form-component_form

---

form {#form}

Last updated: 2021-05-09

Path: miniprogram_gcash

form

2021-05-09 18:43

Form, used to submit the user entry textarea, switch, input, checkbox-group, slider, radio-group, picker and other components in the component.

Clicking the button component with “form” form and “form-type” as “submit” causes submission of the “value” value in the form component. It is required to add “name” in the form component as the key.

Scan QR code to try:

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | onSubmit | EventHandle | Carrying data in form triggers submit event, event.detail = {value : {'name': 'dao14'}, buttonTarget: {'dataset': 'buttonDataset'} } . | | onReset | EventHandle | Trigger reset event upon form reset. |

Screenshot

Sample Code

copy

```
<form onSubmit="formSubmit" onReset="formReset">
  <view class="section section_gap">
```

```
    <view class="section__title">switch</view>
    <switch name="switch"/>
  </view>
  <view class="section section_gap">
    <view class="section__title">slider</view>
    <slider name="slider" show-value ></slider>
  </view>
  <view class="section">
    <view class="section__title">input</view>
    <input name="input" placeholder="please input here" />
  </view>
  <view class="section section_gap">
    <view class="section__title">radio</view>
    <radio-group name="radio-group">
      <label><radio value="radio1"/>radio1</label>
      <label><radio value="radio2"/>radio2</label>
    </radio-group>
  </view>
  <view class="section section_gap">
    <view class="section__title">checkbox</view>
    <checkbox-group name="checkbox">
      <label><checkbox value="checkbox1"/>checkbox1</label>
      <label><checkbox value="checkbox2"/>checkbox2</label>
    </checkbox-group>
  </view>
  <view class="btn-area">
    <button formType="submit">Submit</button>
    <button formType="reset">Reset</button>
  </view>
</form>
```

copy

```
Page({
  formSubmit: function(e) {
    console.log('form has a submit event, carrying data ',
e.detail.value)
  },
  formReset: function() {
    console.log('form has a reset event')
  }
})
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/component_form-component_form

---

getApp {#getapp}

Last updated: 2022-07-03

Path: miniprogram_gcash

getApp

2022-07-03 18:44

A global `getApp()` function is available for obtaining the instance of currently running Mini Program. This is generally used in `page` to get the top-level `app`.

copy

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

Note:

- Do not call `getApp()` in `App()`. Instead, use `this` to get the app instance.
- After the instance is obtained with getApp(), do not call the lifecycle function of `App`.
- Please distinguish `App` global data and `Page` global data.

The global data can be set in App(). The individual sub-pages can get the global application instance through the global function `getApp()`. Here is an example.

copy

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

copy

```
// a.js
// localValue effective only in a.js
var localValue = 'a'
// generating app instance
var app = getApp()
// get global data and change it
app.globalData++
```

copy

```
// b.js
// localValue effective only in b.js
var localValue = 'b'
// if a.js runs first, the globalData returns 2
console.log(getApp().globalData)
```