小程序开发基础

简易教程：https://developers.weixin.qq.com/miniprogram/dev/

申请账号：https://mp.weixin.qq.com/wxopen/waregister?action=step1

开发工具：https://developers.weixin.qq.com/miniprogram/dev/devtools/download.html

基础

小程序运行在微信 APP 中，因此可以借助微信实现普通网页做不到的功能。

小程序并非是 HTML/CSS/JS 体系，因此 jquery 、BOM 和 DOM 是不支持的。采用了 WXML/WXSS/JS 体系，在用法上与前者相似。

一个邮箱只能注册一个小程序。

企业、政府、媒体、其他组织主体可以注册50个小程序，个体户和个人类型主体可注册5个小程序。

每个小程序有自己 AppID 和 AppSecret。

官方提供了几个小程序来辅助开发：小程序助手、小程序数据助手、小程序示例、小程序教学助手和公众平台助手。

公众号关联小程序后，将可在图文消息、自定义菜单、模板消息等功能中使用小程序。

关联规则：

1. 所有公众号都可以关联小程序。
2. 一个公众号可关联10个同主体的小程序，3个不同主体的小程序。
3. 一个小程序可关联500个公众号。
4. 公众号一个月可新增关联小程序13次，小程序一个月可新增关联500次。

ES6

小程序中的 js 是实现了 ECMAScript 6 标准的，为了兼容老设备，需要勾选 ES6 转 ES5，编程时，可以直接使用 ES6 语法（http://es6.ruanyifeng.com/）。

但要特别注意的是，如果勾选了 ES6 转 ES5，那 import 命令将会转换为 require，也就意味着没法支持循环 import，比如 A import B，B import A，就会造成报错（无限递归）。

App

一个小程序就是一个 App，一个 App 可以包含多个页。

基本代码如下：

```
App({
  onLaunch: function(options) {
    // Do something initial when launch.
  },
  onShow: function(options) {
      // Do something when show.
  },
  onHide: function() {
      // Do something when hide.
```

```
  },
  onError: function(msg) {
    console.log(msg)
  },
  globalData: 'I am global data'
})
```

从小程序基础库版本 2.4.0 开始，小程序在手机上支持屏幕旋转。使小程序中的页面支持屏幕旋转的方法是：在 `app.json` 的 `window` 段中设置 `"pageOrientation": "auto"` ，或在页面 json 文件中配置 `"pageOrientation": "auto"` 。

从小程序基础库版本 2.5.0 开始， `pageOrientation` 还可以被设置为 `landscape` ，表示固定为横屏显示。

Page 页

一个页用一个目录表示。该目录下有 .js ， .json，.wxml，.wxss 文件，分别对应业务逻辑、配置文件、页面布局、样式文件，几个文件的文件名必须相同。

业务逻辑基本代码：

```
//获取 App 实例
const app = getApp()

Page({
  data: { // 参与页面渲染的数据
    logs: []
  },
  onLoad: function () {
    // 页面渲染后执行
  }
})
```

WXML 组件

组件列表

视图容器(View Container):

组件名	说明
view	视图容器
scroll-view	可滚动视图容器
swiper	滑块视图容器
movable-view/movable-area	可移动的视图容器
cover-view	覆盖在原生组件之上的文本视图
cover-image	覆盖在原生组件之上的图片视图

基础内容(Basic Content):

组件名	说明
icon	图标
text	文字
rich-text	富文本
progress	进度条

表单(Form)：

标签名	说明
button	按钮
checkbox	多项选择器
form	表单
input	输入框
label	标签
picker	列表选择器
picker-view	内嵌列表选择器
radio	单项选择器
slider	滚动选择器
switch	开关选择器
textarea	多行输入框

导航(Navigation)：

组件名	说明
navigator	页面链接
functional-page-navigator	跳转到插件功能页

多媒体(Media)：

组件名	说明
audio	音频
image	图片
video	视频
camera	系统相机
live-player	实时音视频播放
live-pusher	实时音视频录制

地图(Map):

组件名	说明
map	地图

画布(Canvas):

组件名	说明
canvas	画布

开放能力(Open Ability):

组件名	说明
open-data	展示微信开放的数据
web-view	承载网页的容器
ad	广告
official-account	关注公众号

组件中有些是原生组件，其实就是页面做不到的某些功能，包含：

- `camera`
- `canvas`
- `input`（仅在focus时表现为原生组件）
- `live-player`
- `live-pusher`
- `map`
- `textarea`
- `video`

由微信在 webview 上面叠加的一层，其他组件只能在它之下。

节点信息查询 API

可以用于获取节点属性、样式、在界面上的位置等信息。

最常见的用法是使用这个接口来查询某个节点的当前位置，以及界面的滚动位置。

示例代码：

```
const query = wx.createSelectorQuery()
query.select('#the-id').boundingClientRect(function (res) {
  res.top // #the-id 节点的上边界坐标（相对于显示区域）
})
query.selectViewport().scrollOffset(function (res) {
  res.scrollTop // 显示区域的竖直滚动位置
})
query.exec()
```

上述示例中， `#the-id` 是一个节点选择器，与 CSS 的选择器相近但略有区别，请参见 SelectorQuery.select 的相关说明。

数据绑定

WXML 页面与对应 js 进行数据交互的方式。

基础

```
<!--wxml-->
<view> {{message}} </view>
// page.js
Page({
  data: {
    message: 'Hello MINA!'
  }
})
```

```
<!--wxml-->
<view wx:for="{{array}}"> {{item}} </view>
// page.js
Page({
  data: {
    array: [1, 2, 3, 4, 5]
  }
})
```

条件

```
<!--wxml-->
<view wx:if="{{view == 'WEBVIEW'}}"> WEBVIEW </view>
<view wx:elif="{{view == 'APP'}}"> APP </view>
<view wx:else="{{view == 'MINA'}}"> MINA </view>
// page.js
Page({
  data: {
    view: 'MINA'
  }
})
```

循环

使用 wx:for-item 可以指定数组当前元素的变量名，使用 wx:for-index 可以指定数组当前下标的变量名，下例是九九乘法表：

```
<view wx:for="{{[1, 2, 3, 4, 5, 6, 7, 8, 9]}}" wx:for-item="i">
  <view wx:for="{{[1, 2, 3, 4, 5, 6, 7, 8, 9]}}" wx:for-item="j">
    <view wx:if="{{i <= j}}">
      {{i}} * {{j}} = {{i * j}}
    </view>
  </view>
</view>
```

模板

为了重用布局。

```
<!--定义-->
<template name="staffName">
  <view>
    FirstName: {{firstName}}, LastName: {{lastName}}
  </view>
</template>
<!--使用-->
<template is="staffName" data="{{...staffA}}"></template>
// page.js
Page({
  data: {
    staffA: {firstName: 'Hulk', lastName: 'Hu'}
  }
})
```

扩展运算符 ... 来将一个对象展开。

事件

由“bind+事件类型名”构成，示例如下：

```
<view bindtap="clickMe"> Click me! </view>
Page({
  clickMe: function(event) {
    console.log(event)
  }
})
```

bind 事件绑定不会阻止冒泡事件向上冒泡，catch 事件绑定可以阻止冒泡事件向上冒泡。

Flex 布局

如果你的小程序要求兼容到iOS8以下版本，需要开启样式自动补全。开启样式自动补全，在“项目”—“项目设置”—勾选“上传代码时样式自动补全”。

布局的传统解决方案，基于盒状模型，依赖 display 属性 + position属性 + float属性。它对于那些特殊布局非常不方便，比如，垂直居中就不容易实现。

2009年，W3C 提出了一种新的方案----Flex 布局，可以简便、完整、响应式地实现各种页面布局。目前，它已经得到了所有浏览器的支持，这意味着，现在就能很安全地使用这项功能。

微信提供了 rpx 单位，来适应不同手机屏幕大小，规定屏幕宽为 750rpx。

下面两个参考文档：

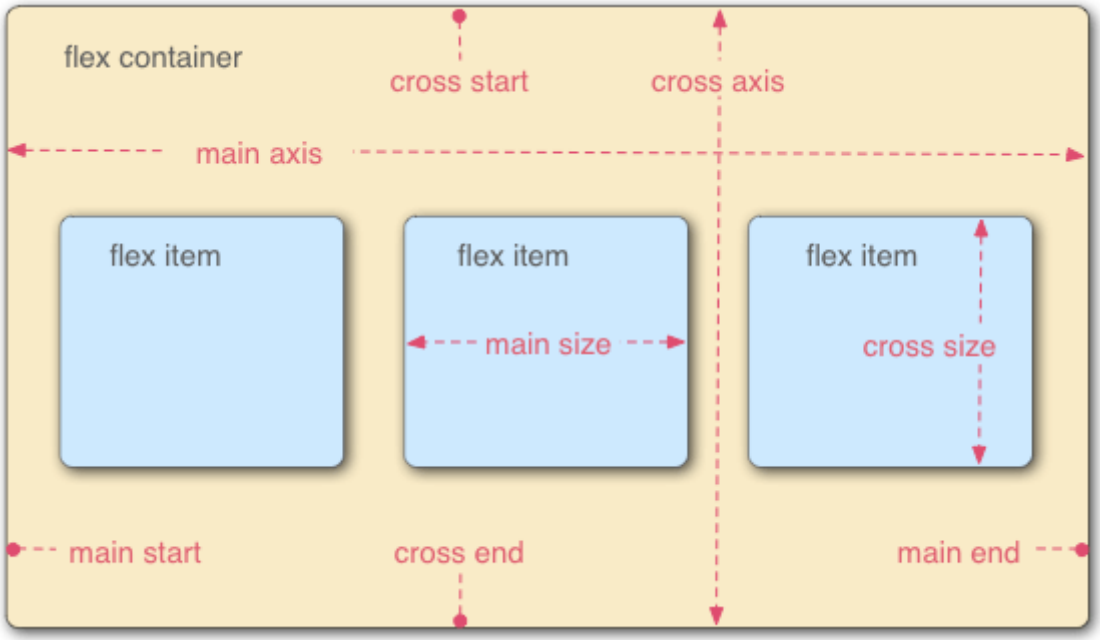
https://developers.weixin.qq.com/ebook?action=get_post_info&docid=00080e799303986b0086e605f5680a

http://www.ruanyifeng.com/blog/2015/07/flex-grammar.html

在线演练：

https://codepen.io/justd/full/yydezN/

基本概念：



这里与传统的概念最大的不同，就是主轴和交叉轴，默认水平为主轴，垂直为交叉轴，但是可以通过 flex-direction 更改，更改完了之后，后续的熟悉设置均依照新的轴向来。而传统的概念中，x/y 轴是不能被改变的。

一旦设置容器为 flex 布局，子元素的 float、clear 和 vertical-align 属性将失效。

而且容器的 align-items、align-content 设置了非 stretch 的话（比如居中），子元素的宽高就是内容大小（表现为非块级元素）。

容器属性

display:flex;

flex-direction:row（默认值） | row-reverse | column |column-reverse，主轴方向

flex-wrap:nowrap（默认值） | wrap | wrap-reverse，依照主轴

justify-content:flex-start（默认值） | flex-end | center |space-between | space-around | space-evenly，依照主轴

align-items:stretch（默认值） | center | flex-end | baseline | flex-start，依照交叉轴

align-content:stretch（默认值） | flex-start | center |flex-end | space-between | space-around | space-evenly，在主轴上有多行（也称为多条轴线），再依照交叉轴

项目属性

order:0（默认值） |

flex-shrink:1（默认值） | ，缩小比例

flex-grow:0（默认值） | ，放大比例

flex-basis:auto（默认值） |

flex:none | auto | @flex-grow @flex-shrink @flex-basis

align-self:auto（默认值） | flex-start | flex-end |center | baseline| stretch

要很好的使用 shrink 和 grow，最佳做法是在兄弟元素都采用默认值，只有一个元素采用设定值，比如三个元素其中两个的 grow 都是 0，而中间的那个 grow 为 1，则中间元素充满剩余空间。

flex-basis 属性定义了在分配多余空间之前，项目占据的主轴空间（main size）。浏览器根据这个属性，计算主轴是否有多余空间。它的默认值为 auto，即项目的本来大小。

flex:1 表达的是 1 1 0%，即放大和缩小比例都是 1，且不占用多余的主轴空间，其意思就是所有孩子按照主轴都保持一样宽或高（而忽略孩子的内容），主轴为横向时就是等宽，为竖向时就是等高。注意这个属性是用到项目上，而不是容器上。

becool.wxss

flex 布局比较恶心的地方在于名词用得太浑。

为了简单起见，我们封装了一套用于 flex 布局的样式。

组件开发

开发者可以将页面内的功能模块抽象成自定义组件，以便在不同的页面中重复使用；也可以将复杂的页面拆分成多个低耦合的模块，有助于代码维护。自定义组件在使用时与基础组件非常相似。

开发

类似于页面，一个自定义组件由 `json` `wxml` `wxss` `js` 4个文件组成。要编写一个自定义组件，首先需要在 `json` 文件中进行自定义组件声明（将 `component` 字段设为 `true` 可这一组文件设为自定义组件）。

在组件 wxss 中不应使用ID选择器、属性选择器和标签名选择器。

在自定义组件的 `js` 文件中，需要使用 `Component()` 来注册组件，并提供组件的属性定义、内部数据和自定义方法。

```
Component({
  properties: {
    // 这里定义了innerText属性，属性值可以在组件使用时指定
    innerText: {
      type: String,
      value: 'default value',
    }
  },
  data: {
    // 这里是一些组件内部数据
    someData: {}
  },
  methods: {
    // 这里是一个自定义方法
    customMethod() {}
```

```
  }
})
```

在组件模板中可以提供一个 `<slot>` 节点，用于承载组件引用时提供的子节点。

```
<!-- 组件模板 -->
<view class="wrapper">
  <view>这里是组件的内部节点</view>
  <slot></slot>
</view>
<!-- 引用组件的页面模板 -->
<view>
  <component-tag-name>
    <!-- 这部分内容将被放置在组件 <slot> 的位置上 -->
    <view>这里是插入到组件slot中的内容</view>
  </component-tag-name>
</view>
```

使用

首先要在页面的 `json` 文件中进行引用声明：

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

一些需要注意的细节：

- 因为 WXML 节点标签名只能是小写字母、中划线和下划线的组合，所以自定义组件的标签名也只能包含这些字符。
- 自定义组件也是可以引用自定义组件的，引用方法类似于页面引用自定义组件的方式（使用 `usingComponents` 字段）。
- 自定义组件和页面所在项目根目录名不能以“wx-”为前缀，否则会报错。

如果你希望这些可复用的代码在多个工程中都可以使用，那最好做成插件，不然就得在每个工程中拷贝一份了。

插件开发

插件是对一组 js 接口、自定义组件或页面的封装（很显然，插件开发包含了组件开发在内），用于嵌入到小程序中使用。插件不能独立运行，必须嵌入在其他小程序中才能被用户使用；而第三方小程序在使用插件时，也无法看到插件的代码。因此，插件适合用来封装自己的功能或服务，提供给第三方小程序进行展示和使用。

插件开发者可以像开发小程序一样编写一个插件并上传代码，在插件发布之后，其他小程序方可调用。小程序平台会托管插件代码，其他小程序调用时，上传的插件代码会随小程序一起下载运行。

只有企业、媒体、政府及其他组织主体的小程序才能开发插件，主体类型为个人的小程序不能开发插件，但可以使用插件。**每个小程序 AppID 只能创建一个插件。**

插件内可以使用全局样式，而插件内的样式则无法外抛出来给他人使用，这就限制了插件本身的能力。

开发

新建插件类型的项目后，如果创建示例项目，则项目中将包含三个目录：

- `plugin` 目录：插件代码目录。
- `miniprogram` 目录：放置一个小程序，用于调试插件。
- `doc` 目录：用于放置插件开发文档。

一个插件可以包含若干个自定义组件、页面，和一组 js 接口。

插件的目录内容如下：

```
plugin
├── components
|   ├── hello-component.js   // 插件提供的自定义组件（可以有多个）
|   ├── hello-component.json
|   ├── hello-component.wxml
|   └── hello-component.wxss
├── pages
|   ├── hello-page.js        // 插件提供的页面（可以有多个，自小程序基础库版本 2.1.0 开始支持）
|   ├── hello-page.json
|   ├── hello-page.wxml
|   └── hello-page.wxss
├── index.js                 // 插件的 js 接口
└── plugin.json              // 插件配置文件
```

可以使用开发者工具打开 `README.md`，并在编辑器的右下角预览插件文档和单独上传插件文档。发布上传文档后，文档不会立刻发布。此时可以使用帐号和密码登录 管理后台，在 小程序插件 > 基本设置 中预览、发布插件文档。

使用

使用时，修改 app.json：

```
"plugins": {
    "myPlugin": {
        "version": "1.0.0",
        "provider": "wxidxxxxxxxxxxxxxxxx"
    }
}
```

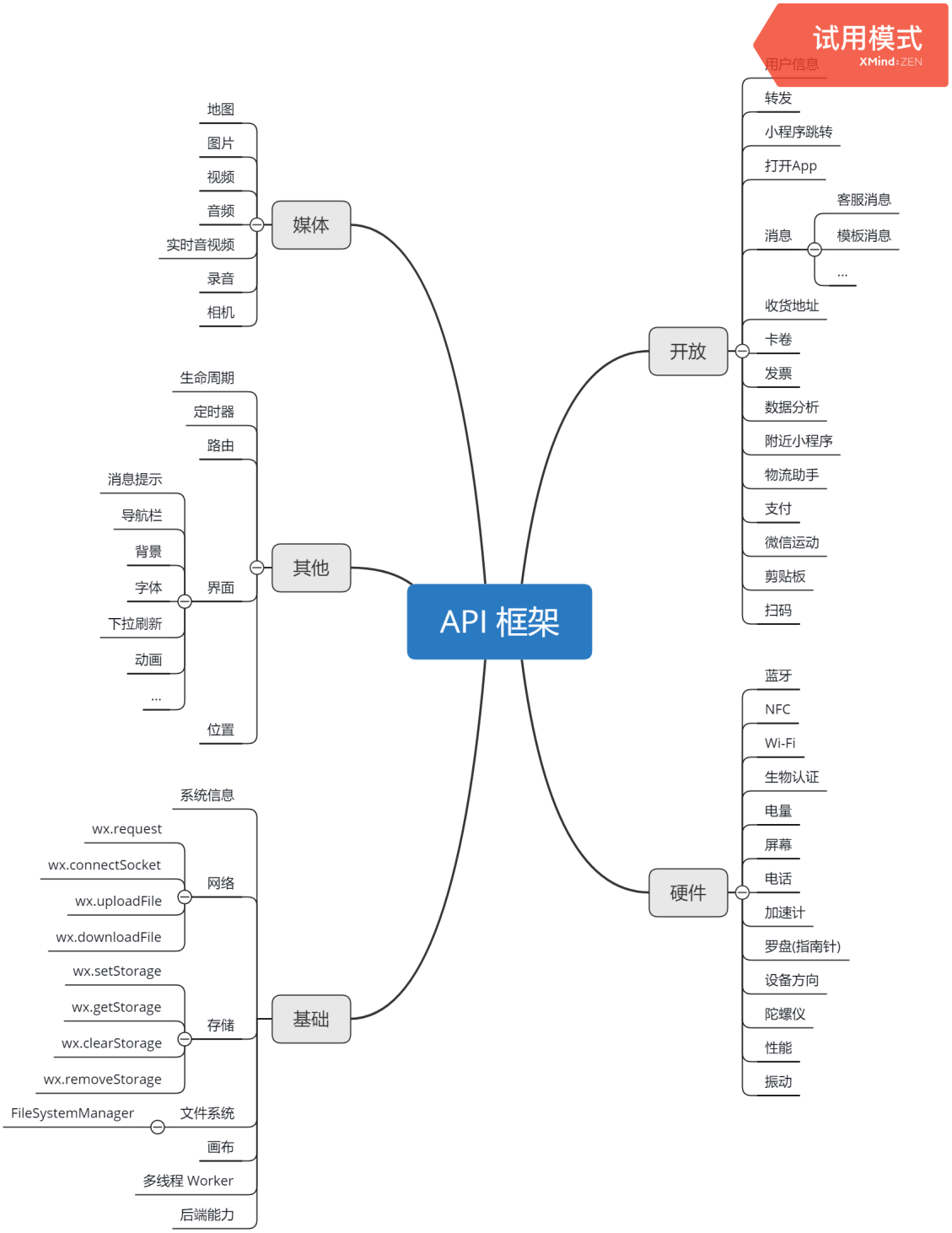
`plugins` 定义段中可以包含多个插件声明，每个插件声明以一个使用者自定义的插件引用名作为标识，并指明插件的 appid 和需要使用的版本号（每次上传插件时指定）。

其中，引用名（如上例中的 `myPlugin`）由使用者自定义，无需和插件开发者保持一致或与开发者协调。

在后续的使用中，该引用名将被用于表示该插件，如：

```
var plugin = requirePlugin("myPlugin");
```

API 框架



与 BeCool 服务器交互

引入 JBeCool

将 BeCool 文件夹放在小程序根目录下。

在你的页面 js 中引入：

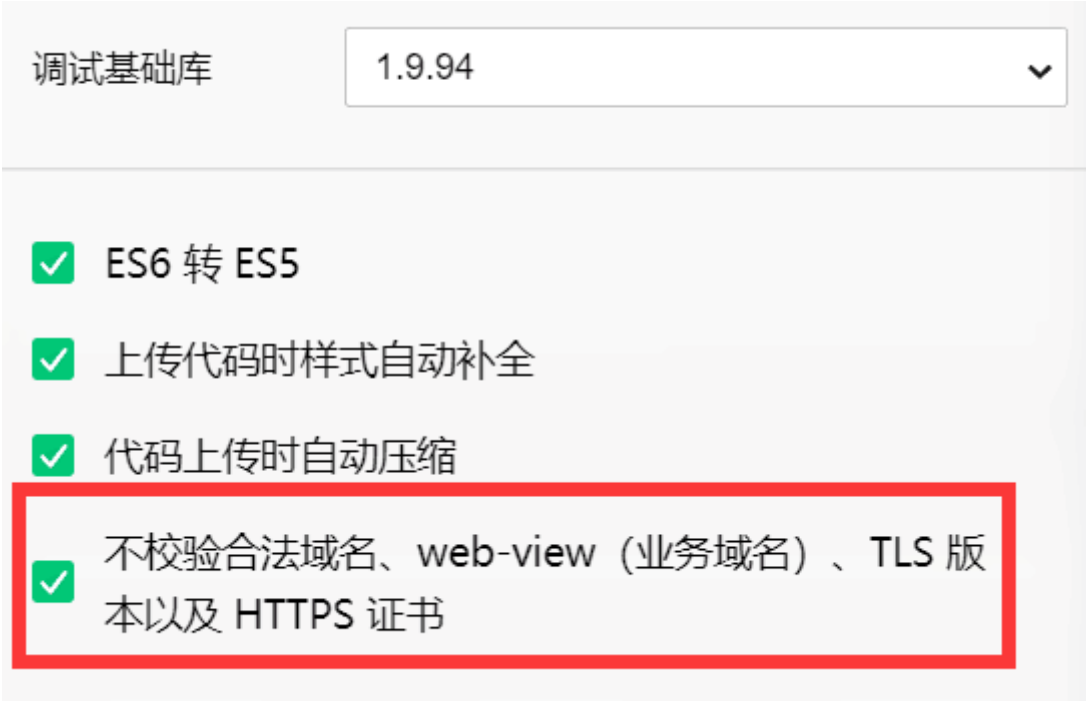
```
var J = require("../../BeCool/JBeCool/wx/imports.js")
```

接下来就可以使用了，如：

```
var cc = J.cc.New('UserId','=',1);
```

发送请求前的配置


上传
测试
腾讯云
详情


调试基础库
1.9.94
ES6 转 ES5
上传代码时样式自动补全
代码上传时自动压缩
不校验合法域名、web-view（业务域名）、TLS 版本以及 HTTPS 证书

发送请求

```
J.send('BeCool.Member.UserControl.GetOne',{id:1},function(u){
    console.log(u.NickName);
});
```

生成小程序码

小程序码本来是唯一的，但可以实现打开小程序时传递一些参数或跳转到特定页面。

为满足不同需求和场景，这里提供了三个接口，开发者可挑选适合自己的接口。

- 接口 A: 适用于需要的码数量较少的业务场景
  - 生成小程序码，可接受 path 参数较长，生成个数受限，数量限制见 注意事项，请谨慎使用。
- 接口 B：适用于需要的码数量极多的业务场景
  - 生成小程序码，可接受页面参数较短，生成个数不受限。
- 接口 C：适用于需要的码数量较少的业务场景

- 生成二维码，可接受 path 参数较长，生成个数受限，数量限制见 注意事项。

注意事项

1. 接口只能生成已发布的小程序的二维码
2. 接口 A 加上接口 C，总共生成的码数量限制为 100,000，请谨慎调用。
3. 接口 B 调用分钟频率受限(5000次/分钟)，如需大量小程序码，建议预生成。

很显然，B 方式更为适合码数极多的情况。

B 方式即时生成参考：https://developers.weixin.qq.com/miniprogram/dev/api/getWXACodeUnlimit.html

这里需要采用服务器端脚本进行调用，如果太多，可以先运行服务器端脚本一次性生成 N 个。

调用时，传递额外参数 scene 进去，就得到一个个小程序码 ，然后用户扫码进来，我们可以通过如下代码获取：

```
Page({
  onLoad: function(options) {
    var scene = decodeURIComponent(options.scene)
  }
})
```

其他资源

通过下面的资源可以了解到，如要构建有价值的小程序技术框架，需要有：

- 云。省去了自己构建服务器的麻烦，而且云端要提供大量基础功能，比如数据库操作、文件操作、图片处理等。
- 组件与插件。提供功能丰富的组件和插件，大大提升小程序开发效率。
- 多端统一。一次开发，可以部署为微信小程序、支付宝小程序、百度智能小程序、H5、APP等等。

在上述诉求下，除了第三点，腾讯官方都做得很好了，故目前的技术框架前景不容乐观。

腾讯云

官方提供的小程序云端服务。

https://developers.weixin.qq.com/miniprogram/dev/wxcloud/basis/getting-started.html

https://cloud.tencent.com/solution/la

知晓云

提供快速小程序开发的框架，并提供了云端，免去了自己搭建的麻烦。类似腾讯云。

https://cloud.minapp.com/

WePY

一个最受欢迎的小程序框架。

https://tencent.github.io/wepy/

Tina.js

一款轻巧的渐进式微信小程序框架。

https://github.com/tinajs/tina

Taro

一套遵循 React 语法规范的多端统一开发框架。一次开发，多端部署。

https://taro.aotu.io/

weweb

weweb是一个兼容小程序语法的前端框架，你可以用小程序的写法，来写web应用。

跨平台，一套代码多端运行（小程序、h5、未来还可以直接打包成app)。

WXPage

WXPage 是一个极其轻量的微信小程序开发框架，其中的API蕴含了“极致页面打开速度的思想”，为可维护性与开发效率而设计的功能，框架起源于“腾讯视频”小程序。

https://github.com/tvfe/wxpage

UI组件、开发框架、实用库、开发工具

https://www.cnblogs.com/icyhoo/p/6282574.html

http://jcodecraeer.com/a/qianduankaifa/2018/0418/9611.html

https://www.91ud.com/app/23879.html（插件商店）

https://minapp.com/miniapp/（小程序商店、插件市场）

https://developers.weixin.qq.com/community/plugins（腾讯官方插件市场）

WeUI

WeUI 是一套同微信原生视觉体验一致的基础样式库，由微信官方设计团队为微信内网页和微信小程序量身设计，令用户的使用感知更加统一。

https://github.com/Tencent/weui-wxss