公众帐号：是平台唯一的帐号，只能用于登录公众平台，而不能登录手机微信。

管理员微信号：是属于个人微信号，不能用于登录公众平台。但绑定后，其在平台的权限是最大的，可以操作所有功能。

运营者微信号：也是属于个人微信号，也不能用于登录公众平台。运营者微信号权限相对少，算是是管理员的下属吧。

运营者的授权来自管理员，要求运营者先关注微信公众号。

计算机学院不止一个公众号，“川大计算机2015级”需要准备材料，然后去学院盖章。

在微信里面，一个公众号的用户都会对应的存在一个openid。同一个用户在不同的公众号下面的openid都不一样，但是对于某个指定的公众号，这个用户的openid是固定的。openid本身不携带任何用户的个人信息，但是可以用来确认用户个体的唯一性。使用snsapi\_base为scope的授权方式可以在不打扰用户的情况下获取到用户的openid，详细需要见开发文档。你可以简单的理解为

openid = hash(uid + app\_id)也就是说，对每个app可以做到用户id唯一，但是这个id又不是通用的，你在其它app上同样的用户，openid会完全不同，反正你无法通过这个id获取用户的任何信息。用户关注公众号前后openid不会改变。

获取openid，要么是网页OAuth2授权，要么就是在关注或者与公众号发生交互的时候。所以如果因为公众号过期导致不能用网页授权的方式来获取openid，就需要配合公众号了！与公众号进行交互返回wwww.xxx.cn?openid=xxxx当然不要叫openid也好，免得被发现什么。一旦引发公众号交互并借此在数据库中插入openid，网页访问的时候先检验openid是否存在，体育理论考试的15次就可以这么做与openid捆绑。

用户管理类接口中的“获取用户基本信息接口”，是在用户和公众号产生消息交互或关注后事件推送后，才能根据用户OpenID来获取用户基本信息。这个接口，包括其他微信接口，都是需要该用户（即openid）关注了公众号后，才能调用成功的。

微信web：

请使用微信扫描二维码登录“微信web开发者工具”

1. 你可以使用自己的微信号来调试微信网页授权;

2. 你可以调试、检验页面的JS-SDK相关功能与权限，模拟大部分 SDK 的输入和输出;

3. 你可以使用 X5 Blink 调试，或者是基于 weinre 的移动调试功能;

4. 你可以利用集成的Chrome DevTools协助开发（chrome的F12）。

这个工具有一个缺点，内置的浏览器只能访问http或者https协议，不能访问本机html文件也就是file:///，所以我做完以后还要上传到自己的服务器才能查看到效果。

为了保证开发者身份信息的安全，对于希望调试的公众号，我们要求开发者微信号与之建立绑定关系。具体操作为：公众号登录管理后台，启用开发者中心，在开发者工具——web 开发者工具页面，向开发者微信号发送绑定邀请。每个公众号最多可同时绑定10个开发者微信号。完成登录和绑定后，开发者就可以开始调试微信网页授权和群发操作了，注意只能调试自己绑定过的公众号。我没有设定自己的微信号，但是可以用手机号来代替。

在右侧的 JS-SDK Tab 中看到当前页面 wx.config 的校验情况和 JSSDK 的调用 log。如果校验未通过的话在config那里type就会使error。

在右侧的权限列表，可以查看到通过 wx.config 获取到的权限。

移动调试功能是基于 weinre 的，并且做了一些改进，相比直接使用 weinre 有两个优点：

无须手工在页面中加入 weinre 调试脚本

可以在 weinre 的网络请求页卡中，看到完整的 http 请求 log，非局限于 ajax 请求

OAuth : 目前协议最新的版本是2.0，协议的存在是为了第三方应用来服务器上获取资源的同时保证用户的账户密码得到保障

Resource Owner：资源所有者，本文中又称"用户"（user）。

Resource server：资源服务器，即服务提供商存放用户生成的资源的服务器。它与认证服务器，可以是同一台服务器，也可以是不同的服务器。

OAuth在"客户端"与"服务提供商"之间，设置了一个授权层（authorization layer）。"客户端"不能直接登录"服务提供商"，只能登录授权层，以此将用户与客户端区分开来。"客户端"登录授权层所用的令牌（token），与用户的密码不同。用户可以在登录的时候，指定授权层令牌的权限范围和有效期。

（A）用户打开客户端以后，客户端要求用户给予授权。

（B）用户同意给予客户端授权。

（C）客户端使用上一步获得的授权，向认证服务器申请令牌。

（D）认证服务器对客户端进行认证以后，确认无误，同意发放令牌。

（E）客户端使用令牌，向资源服务器申请获取资源。

（F）资源服务器确认令牌无误，同意向客户端开放资源。

（B）是最难的。客户端必须得到用户的授权（authorization grant），才能获得令牌（access token）。OAuth 2.0定义了四种授权方式。

授权码模式（authorization code）

简化模式（implicit）

密码模式（resource owner password credentials）

客户端模式（client credentials）

授权码模式（authorization code）是功能最完整、流程最严密的授权模式。它的特点就是通过客户端的后台服务器，与"服务提供商"的认证服务器进行互动。

（A）用户访问客户端，后者将前者导向认证服务器。

（B）用户选择是否给予客户端授权。

（C）假设用户给予授权，认证服务器将用户导向客户端事先指定的"重定向URI"（redirection URI），同时附上一个授权码。

（D）客户端收到授权码，附上早先的"重定向URI"，向认证服务器申请令牌。这一步是在客户端的后台的服务器上完成的，对用户不可见。用户可见授权码，但是见不到令牌。

（E）认证服务器核对了授权码和重定向URI，确认无误后，向客户端发送访问令牌（access token）和更新令牌（refresh token）。

A步骤中，客户端申请认证的URI，包含以下参数：（在微信里面就是授权回调网址）

response\_type：表示授权类型，必选项，此处的值固定为"code"

client\_id：表示客户端的ID，必选项

redirect\_uri：表示重定向URI，可选项

scope：表示申请的权限范围，可选项

state：表示客户端的当前状态，可以指定任意值，认证服务器会原封不动地返回这个值。

C步骤中，服务器回应客户端的URI，包含以下参数：

code：表示授权码，必选项。该码的有效期应该很短，通常设为10分钟，客户端只能使用该码一次，否则会被授权服务器拒绝。该码与客户端ID和重定向URI，是一一对应关系。

state：如果客户端的请求中包含这个参数，认证服务器的回应也必须一模一样包含这个参数。

D步骤中，客户端向认证服务器申请令牌的HTTP请求，包含以下参数：

grant\_type：表示使用的授权模式，必选项，此处的值固定为"authorization\_code"。

code：表示上一步获得的授权码，必选项。

redirect\_uri：表示重定向URI，必选项，且必须与A步骤中的该参数值保持一致。

client\_id：表示客户端ID，必选项。

E步骤中，认证服务器发送的HTTP回复，包含以下参数：

access\_token：表示访问令牌，必选项。

token\_type：表示令牌类型，该值大小写不敏感，必选项，可以是bearer类型或mac类型。

expires\_in：表示过期时间，单位为秒。如果省略该参数，必须其他方式设置过期时间。

refresh\_token：表示更新令牌，用来获取下一次的访问令牌，可选项。

scope：表示权限范围，如果与客户端申请的范围一致，此项可省略。

其实E步骤可能存在DDOS漏洞。

简化模式就是取消了授权码，用户浏览器直接向认证服务器申请令牌，那么令牌也就是用户可见的。

如果用户在微信客户端中访问第三方的网页，公众号可以通过微信网页授权机制，来获取用户基本信息，进而实现业务逻辑。

OAuth2.0(Open Authorization, OAUTH协议为用户资源的授权提供了一个安全的、开放而又简易的标准。)，在使用中需要配置指定的链接：https://open.weixin.qq.com/connect/oauth2/authorize?appid=APPID&redirect\_uri=REDIRECT\_URI&response\_type=code&scope=SCOPE&state=STATE#wechat\_redirect

在这边链接中的REDIRECT\_URI必须是我们填在微信公众号中的回调域名，我们在打开的时候才不会报错，否则会报redirect\_uri不对的错误，这样就很不方便。

两种网页授权方式：

以snsapi\_base为scope发起的网页授权，是用来获取进入页面的用户的openid的，并且是静默授权并自动跳转到回调页的。用户感知的就是直接进入了回调页（往往是业务页面）

以snsapi\_userinfo为scope发起的网页授权，是用来获取用户的基本信息的。但这种授权需要用户手动同意，并且由于用户同意过，所以无须关注，就可在授权后获取该用户的基本信息。

关于网页授权access\_token和普通access\_token的区别（也就是获取的方式不同）

1、微信网页授权是通过OAuth2.0机制实现的，在用户授权给公众号后，公众号可以获取到一个网页授权特有的接口调用凭证（网页授权access\_token），通过网页授权access\_token可以进行授权后接口调用，如获取用户基本信息；

2、其他微信接口，需要通过基础支持中的“获取access\_token”接口来获取到的普通access\_token调用。

微信推出的网页开发工具包-微信JS-SDK，这js包里面封装着微信提供的接口。你使用js来进行调用。

http://203.195.235.76/jssdk/ 是一个已经写了的示例。

模拟谷歌Android：

chrome.exe –user-agent=”Mozilla/5.0 (Linux; U; Android 2.2; en-us; Nexus One Build/FRF91) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1″

模拟苹果iPhone：

chrome.exe –user-agent=”Mozilla/5.0 (iPhone; CPU iPhone OS 5\_0\_1 like Mac OS X) AppleWebKit/534.46 (KHTML, like Gecko) Mobile/9A405″

对公众号中的网页进行检测，只允许Tecent内置的qq浏览器的访问：

1.使用js来检测UserAgent，但是可以借助防跳转然后删除这段js就好

<script type="text/javascript">

// 对浏览器的UserAgent进行正则匹配，不含有微信独有标识的则为其他浏览器

var useragent = navigator.userAgent;

if (useragent.match(/MicroMessenger/i) != 'MicroMessenger') {

// 这里警告框会阻塞当前页面继续加载

alert('已禁止本次访问：您必须使用微信内置浏览器访问本页面！');

// 以下代码是用javascript强行关闭当前页面

var opened = window.open('about:blank', '\_self');

opened.opener = null;

opened.close();

}

</script>

2.要求访问网页的人进行只有微信客户端才能进行的用户验证，比如说cookie/session，用户openid等

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width,initial-scale=1,user-scalable=0">

<title>xxx</title>

<link rel="stylesheet" href="path/to/weui/dist/style/xxx.min.css"/>

</head>

<body>

</body>

</html>

微信应用程序：

在我们在开发微信第三方应用的时候也使用到了发起网页授权，即代公众号发起网页授权，在文档里也是有一个固定的链接

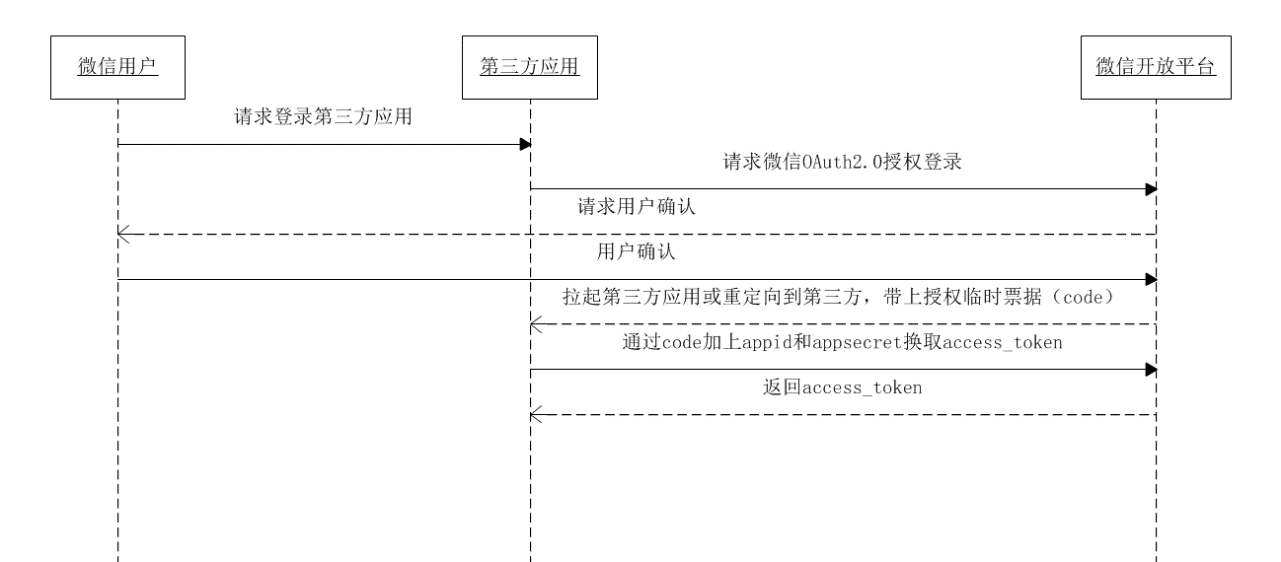
https://open.weixin.qq.com/connect/oauth2/authorize?appid=APPID&redirect\_uri=REDIRECT\_URI&response\_type=code&scope=SCOPE&state=STATE&component\_appid=component\_appid#wechat\_redirect

在这个链接中我们可以看到在结尾多了一个component\_appid（第三方应用的app\_id），当我们使用这个链接时是不需要在微信公众号里配置回调域名的，这样也是能够拿到code，再根据code拿到用户的opendId，这样当我们在点击菜单的时候就能够一步到位拿到用户信息了。

weui.io对weui这个开源js组件库的使用举例

微信Android SDK开发：

在下面流程图中，每一步关键步骤都是“自己”去做的



snsapi\_base可以让移动端网页授权绕过跳转授权登录页请求用户授权的动作，直接跳转第三方网页带上授权临时票据（code），但会使得用户已授权作用域（scope）仅为snsapi\_base，从而导致无法获取到需要用户授权才允许获得的数据和基础功能。这种作用域应该就是打开以下微信客户端知道该返回一个怎么样的code回来，因为除了能登录以外没有然后去权限，不会泄露任何用户信息，所以也就没有用户授权界面。

sendResp 这个函数是第三方应用调用的，结果会让微信来请求，自己成为响应端。