微信小程序入门教程之四：API 使用

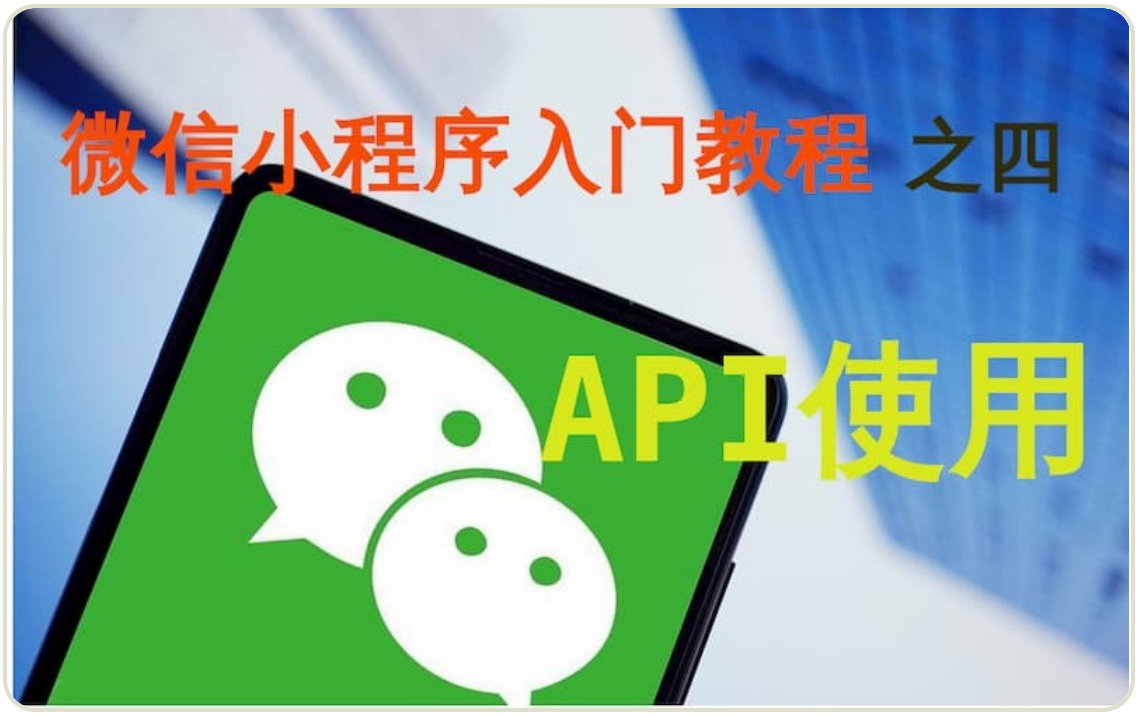
作者： 阮一峰

日期： 2020年11月 2日

今天是这个系列教程的最后一篇。

上一篇教程介绍了，小程序页面如何使用 JavaScript 脚本。有了脚本以后，就可以调用微信提供的各种能力（即微信 API），从而做出千变万化的页面。本篇就介绍怎么使用 API。

所有示例的完整代码，都可以从 GitHub 的代码仓库下载。



一、WXML 渲染语法

前面说过，小程序的页面结构使用 WXML 语言进行描述。

WXML 的全称是微信页面标签语言（Weixin Markup Language），它不仅提供了许多功能标签，还有一套自己的语法，可以设置页面渲染的生效条件，以及进行循环处理。

微信 API 提供的数据，就通过 WXML 的渲染语法展现在页面上。比如，`home.js` 里面的数据源是一个数组。

```
Page({
  data: {
    items: ['事项 A', '事项 B', '事项 C']
  }
});
```

上面代码中，`Page()` 的参数配置对象的 `data.items` 属性是一个数组。通过数据绑定机制，页面可以读取全局变量 `items` ，拿到这个数组。

拿到数组以后，怎样将每一个数组成员展现在页面上呢？WXML 的数组循环语法，就是一个很简便的方法。

打开 `home.wxml` ，改成下面的代码。

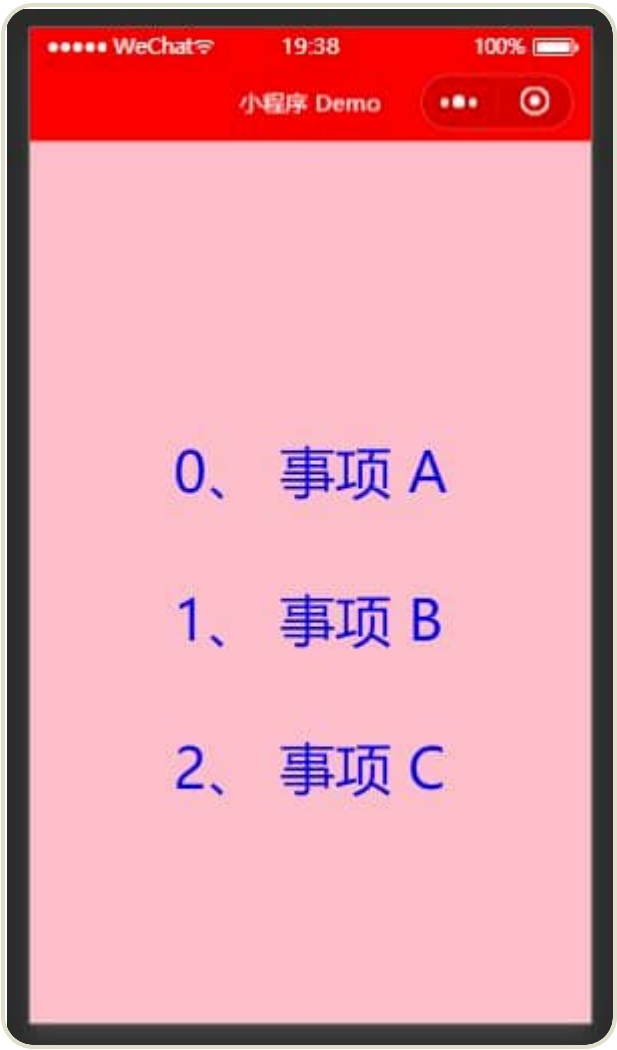
```
<view>
  <text class="title" wx:for="{{items}}">
    {{index}}、 {{item}}
   </text>
</view>
```

上面代码中，`<text>` 标签的 `wx:for` 属性，表示当前标签（ `<text>` ）启用数组循环，处理 `items` 数组。数组有多少个成员，就会生成多少个 `<text>` 。渲染后的页面结构如下。

```
<view>
  <text>...</text>
  <text>...</text>
  <text>...</text>
</view>
```

在循环体内，当前数组成员的位置序号（从 `0` 开始）绑定变量 `index` ，成员的值绑定变量 `item` 。

开发者工具导入项目代码，页面渲染结果如下。



这个示例的完整代码，可以参考代码仓库。

WXML 的其他渲染语法（主要是条件判断和对象处理），请查看官方文档。

二、客户端数据储存

页面渲染用到的外部数据，如果每次都从服务器或 API 获取，有时可能会比较慢，用户体验不好。

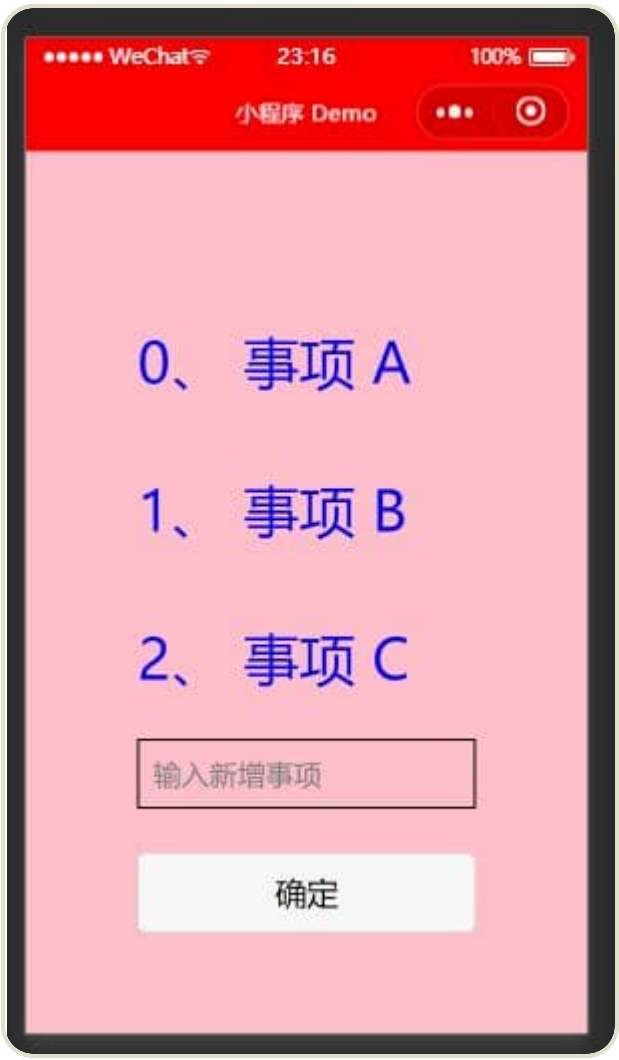
小程序允许将一部分数据保存在客户端（即微信 App）的本地储存里面（其实就是自定义的缓存）。下次需要用到这些数据的时候，就直接从本地读取，这样就大大加快了渲染。本节介绍怎么使用客户端数据储存。

打开 `home.wxml` ，改成下面的代码。

```
<view>
  <text class="title" wx:for="{{items}}">
    {{index}}、 {{item}}
   </text>
   <input placeholder="输入新增事项" bind:input="inputHandler"/>
   <button bind:tap="buttonHandler">确定</button>
</view>
```

上面代码除了展示数组 `items` ，还新增了一个输入框和一个按钮，用来接受用户的输入。背后的意图是，用户通过输入框，为 `items` 数组加入新成员。

开发者工具导入项目代码，页面渲染结果如下。



注意，输入框有一个 `input` 事件的监听函数 `inputHandler` （输入内容改变时触发），按钮有一个 `tap` 事件的监听函数 `buttonHandler` （点击按钮时触发）。这两个监听函数负责处理用户的输入。

然后，打开 home.js ，代码修改如下。

```
Page({
  data: {
    items: [],
    inputValue: ''
  },
  inputHandler(event) {
    this.setData({
      inputValue: event.detail.value || ''
    });
  },
  buttonHandler(event) {
    const newItem = this.data.inputValue.trim();
    if (!newItem) return;
    const itemArr = [...this.data.items, newItem];
    wx.setStorageSync('items', itemArr);
    this.setData({ items: itemArr });
  },
  onLoad() {
    const itemArr = wx.getStorageSync('items') || [];
    this.setData({ items: itemArr });
  }
});
```

上面代码中，输入框监听函数 inputHandler() 只做了一件事，就是每当用户的输入发生变化时，先从事件对象 event 的 detail.value 属性上拿到输入的内容，然后将其写入全局变量 inputValue 。如果用户删除了输入框里面的内容， inputValue 就设为空字符串。

按钮监听函数 buttonHandler() 是每当用户点击提交按钮，就会执行。它先从 inputValue 拿到用户输入的内容，确定非空以后，就将其加入 items 数组。然后，使用微信提供的 wx.setStorageSync() 方法，将 items 数组存储在客户端。最后使用 this.setData() 方法更新一下全局变量 items ，进而触发页面的重新渲染。

wx.setStorageSync() 方法属于小程序的客户端数据储存 API，用于将数据写入客户端储存。它接受两个参数，分别是键名和键值。与之配套的，还有一个 wx.getStorageSync() 方法，用于读取客户端储存的数据。它只有一个参数，就是键名。这两个方法都是同步的，小程序也提供异步版本，请参考官方文档。

最后，上面代码中， `Page()` 的参数配置对象还有一个 `onLoad()` 方法。该方法属于页面的生命周期方法，页面加载后会自动执行该方法。它只执行一次，用于页面初始化，这里的意图是每次用户打开页面，都通过 `wx.getStorageSync()` 方法，从客户端取出以前存储的数据，显示在页面上。

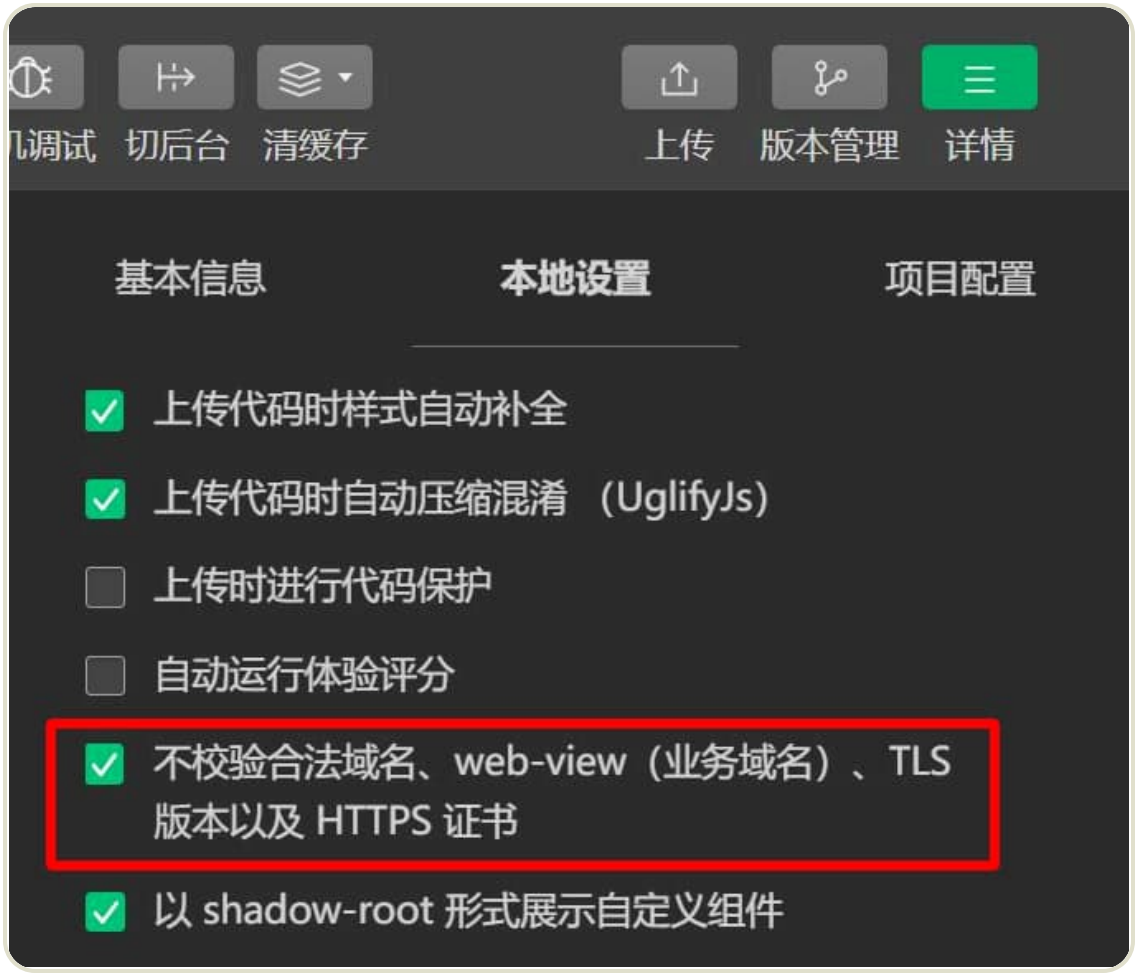
这个示例的完整代码，可以参考代码仓库。

必须牢记的是，客户端储存是不可靠的，随时可能消失（比如用户清理缓存）。用户换了一台手机，或者本机重装微信，原来的数据就丢失了。所以，它只适合保存一些不重要的临时数据，最常见的用途一般就是作为缓存，加快页面显示。

三、远程数据请求

小程序可以从外部服务器读取数据，也可以向服务器发送数据。本节就来看看怎么使用小程序的网络能力。

微信规定，只有后台登记过的服务器域名，才可以进行通信。不过，开发者工具允许开发时放松这个限制。



按照上图，点击开发者工具右上角的三条横线（"详情"），选中"不校验合法域名、web-view（业务域名）、TLS 版本以及 HTTPS 证书"。这样的话，小程序在开发时，就可以跟服务器进行通信了。

下面，我们在本地启动一个开发服务器。为了简单起见，我选用了 json-server 作为本地服务器，它的好处是只要有一个 JSON 数据文件，就能自动生成 RESTful 接口。

首先，新建一个数据文件 `db.json`，内容如下。

```
{
  "items": ["事项 A", "事项 B", "事项 C"]
}
```

然后，确认本机安装了 Node.js 以后，进入 `db.json` 所在的目录，在命令行执行下面命令，启动服务器。

```
npx json-server db.json
```

正常情况下，这时你打开浏览器访问 `localhost:3000/items` 这个网址，就能看到返回了一个数组 `["事项 A", "事项 B", "事项 C"]` 。

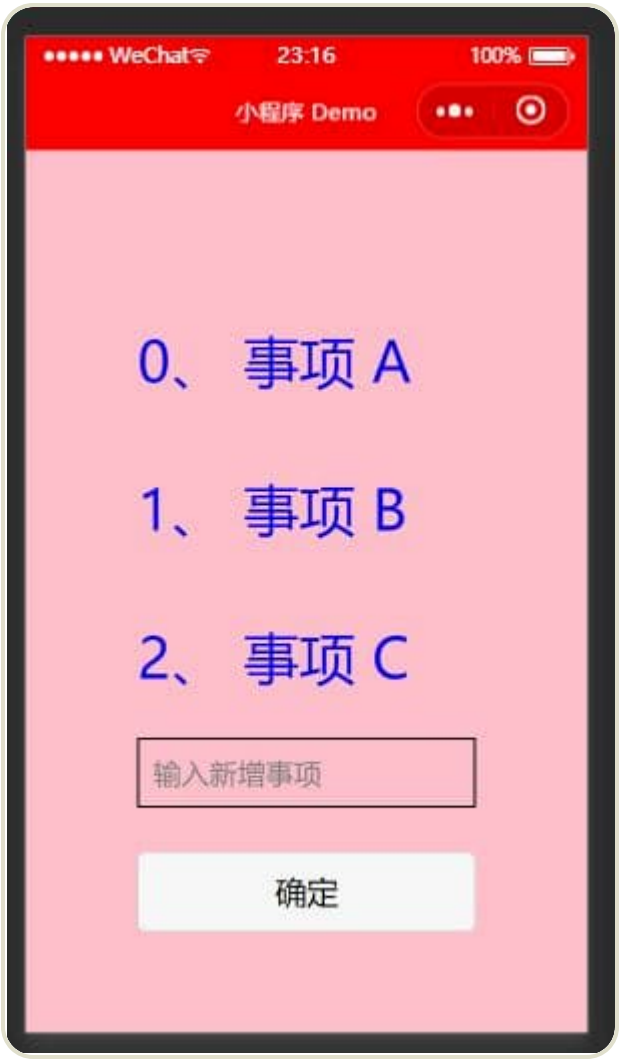
接着，打开 `home.js` ，代码修改如下。

```
Page({
  data: { items: [] },
  onLoad() {
    const that = this;
    wx.request({
      url: 'http://localhost:3000/items',
      success(res) {
        that.setData({ items: res.data });
      }
    });
  }
});
```

上面代码中，生命周期方法 `onLoad()` 会在页面加载后自动执行，这时就会执行 `wx.request()` 方法去请求远程数据。如果请求成功，就会执行回调函数 `succcess()` ，更新页面全局变量 `items` ，从而让远程数据显示在页面上。

`wx.request()` 方法就是小程序的网络请求 API，通过它可以发送 HTTP 请求。它的参数配置对象最少需要指定 `url` 属性（请求的网址）和 `succcess()` 方法（服务器返回数据的处理函数）。其他参数请参考官方文档。

开发者工具导入项目代码，页面渲染结果如下。它的初始数据是从服务器拿到的。



这个示例的完整代码，可以参考代码仓库。

这个例子只实现了远程数据获取，json-server 实际上还支持数据的新增和删改，大家可以作为练习，自己来实现。

四、<open-data>组件

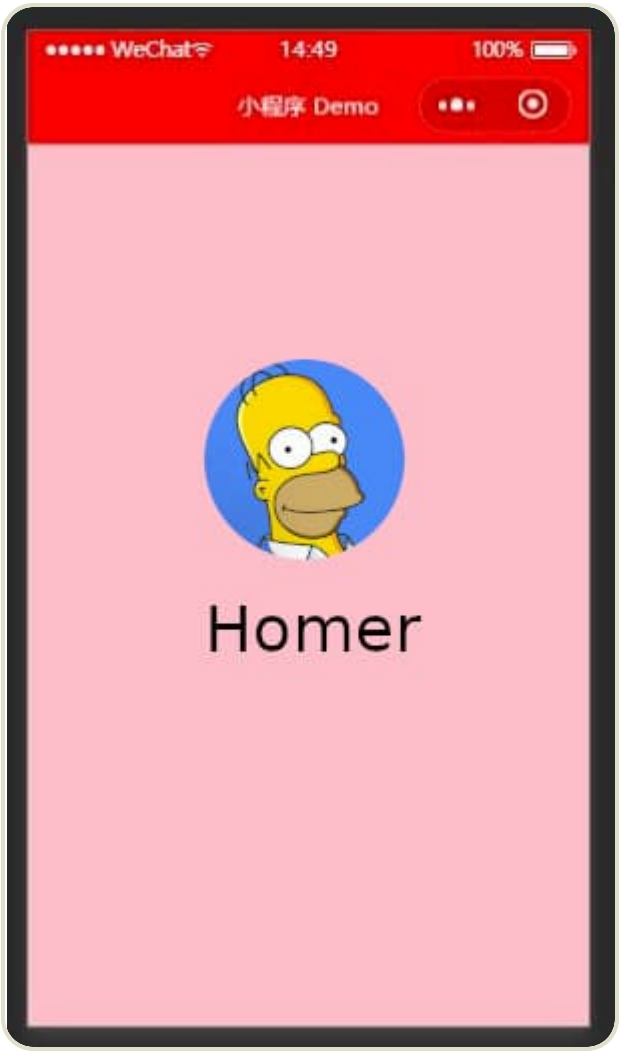
如果要在页面上展示当前用户的身份信息，可以使用小程序提供的 `<open-data>` 组件。

打开 `home.wxml` 文件，代码修改如下。

```
<view>
  <open-data type="userAvatarUrl"></open-data>
  <open-data type="userNickName"></open-data>
</view>
```

上面代码中， `<open-data>` 组件的 `type` 属性指定所要展示的信息类型， `userAvatarUrl` 表示展示用户头像， `userNickName` 表示用户昵称。

开发者工具导入项目代码，页面渲染结果如下，显示你的头像和用户昵称。



`<open-data>` 支持的用户信息如下。

- `userNickName`：用户昵称
- `userAvatarUrl`：用户头像
- `userGender`：用户性别
- `userCity`：用户所在城市
- `userProvince`：用户所在省份
- `userCountry`：用户所在国家
- `userLanguage`：用户的语言

这个示例的完整代码，可以参考代码仓库。

`<open-data>` 不需要用户授权，也不需要登录，所以用起来很方便。但也是因为这个原因，小程序不允许用户脚本读取 `<open-data>` 返回的信息。

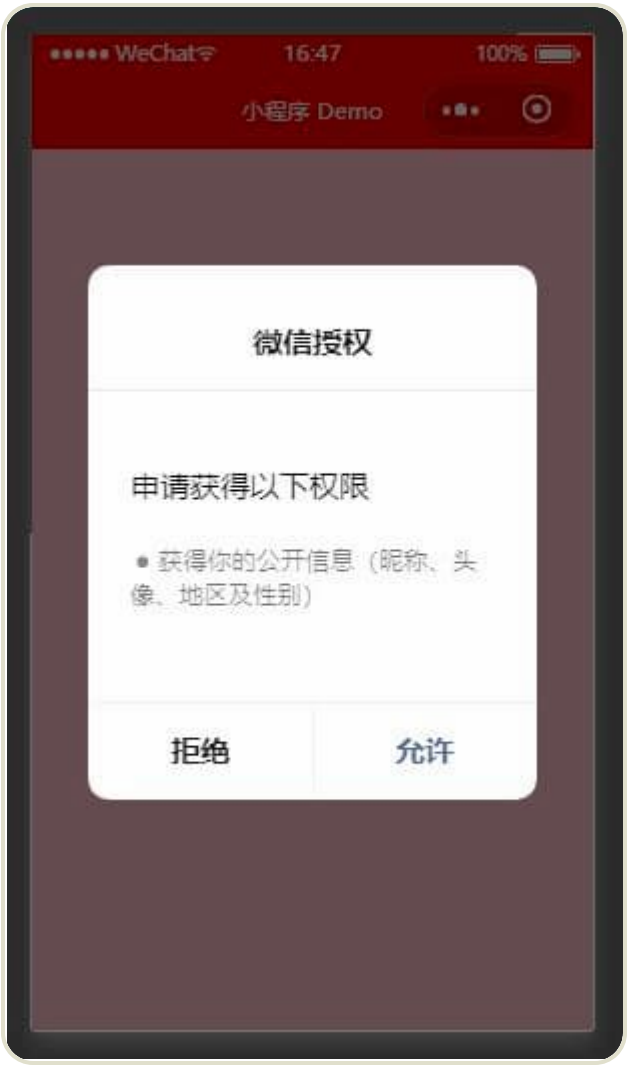
五、获取用户个人信息

如果想拿到用户的个人信息，必须得到授权。官方建议，通过按钮方式获取授权。

打开 `home.wxml` 文件，代码修改如下。

```
<view>
  <text class="title">hello {{name}}</text>
  <button open-type="getUserInfo" bind:getuserinfo="buttonHandler">
    授权获取用户个人信息
  </button>
</view>
```

上面代码中，`<button>` 标签的 `open-type` 属性，指定按钮用于获取用户信息，`bind:getuserinfo` 属性表示点击按钮会触发 `getuserinfo` 事件，即跳出对话框，询问用户是否同意授权。



用户点击"允许"，脚本就可以得到用户信息。

`home.js` 文件的脚本代码如下。

```
Page({
  data: { name: '' },
  buttonHandler(event) {
    if (!event.detail.userInfo) return;
    this.setData({
      name: event.detail.userInfo.nickName
    });
  }
});
```

上面代码中， `buttonHandler()` 是按钮点击的监听函数，不管用户点击"拒绝"或"允许"，都会执行这个函数。我们可以通过事件对象 `event` 有没有 `detail.userInfo` 属性，来判断用户点击了哪个按钮。如果能拿到

`event.detail.userInfo` 属性，就表示用户允许读取个人信息。这个属性是一个对象，里面就是各种用户信息，比如头像、昵称等等。

这个示例的完整代码，可以参考代码仓库。

实际开发中，可以先用 `wx.getSetting()` 方法判断一下，用户是否已经授权过。如果已经授权过，就不用再次请求授权，而是直接用 `wx.getUserInfo()` 方法获取用户信息。

注意，这种方法返回的用户信息之中，不包括能够真正识别唯一用户的 `openid` 属性。这个属性需要用到保密的小程序密钥去请求，所以不能放在前端获取，而要放在后端。这里就不涉及了。

六、多页面的跳转

真正的小程序不会只有一个页面，而是多个页面，所以必须能在页面之间实现跳转。

`app.json` 配置文件的 `pages` 属性就用来指定小程序有多少个页面。

```
{
  "pages": [
    "pages/home/home",
    "pages/second/second"
  ],
  "window": ...
}
```

上面代码中， `pages` 数组包含两个页面。以后每新增一个页面，都必须把页面路径写在 `pages` 数组里面，否则就是无效页面。排在第一位的页面，就是小程序打开时，默认展示的页面。

新建第二个页面的步骤如下。

第一步，新建 `pages/second` 目录。

第二步，在该目录里面，新建文件 `second.js` ，代码如下。

```
Page({});
```

第三步，新建第二页的页面文件 `second.wxml` ，代码如下。

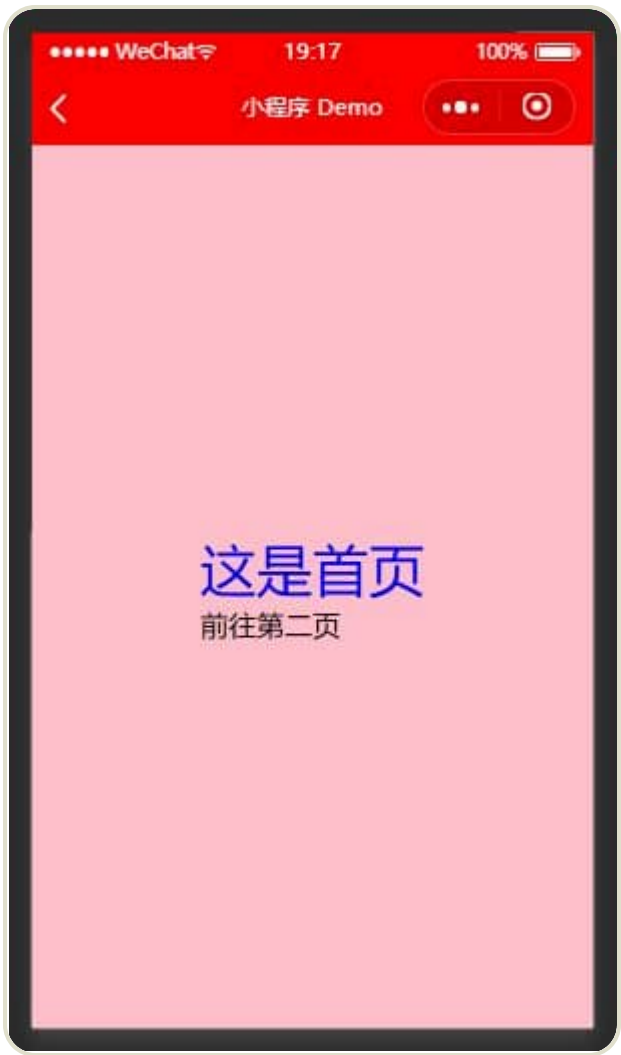
```
<view>
  <text class="title">这是第二页</text>
  <navigator url="../home/home">前往首页</navigator>
</view>
```

上面代码中， `<navigator>` 就是链接标签，相当于网页标签 `<a>` ，只要用户点击就可以跳转到 `url` 属性指定的页面（这里是第一页的位置）。

第四步，修改第一页的页面文件 `home.wxml` ，让用户能够点击进入第二页。

```
<view>
  <text class="title">这是首页</text>
  <navigator url="../second/second">前往第二页</navigator>
</view>
```

开发者工具导入项目代码，页面渲染结果如下。



用户点击"前往第二页"，就会看到第二个页面。

这个示例的完整代码，可以参考代码仓库。

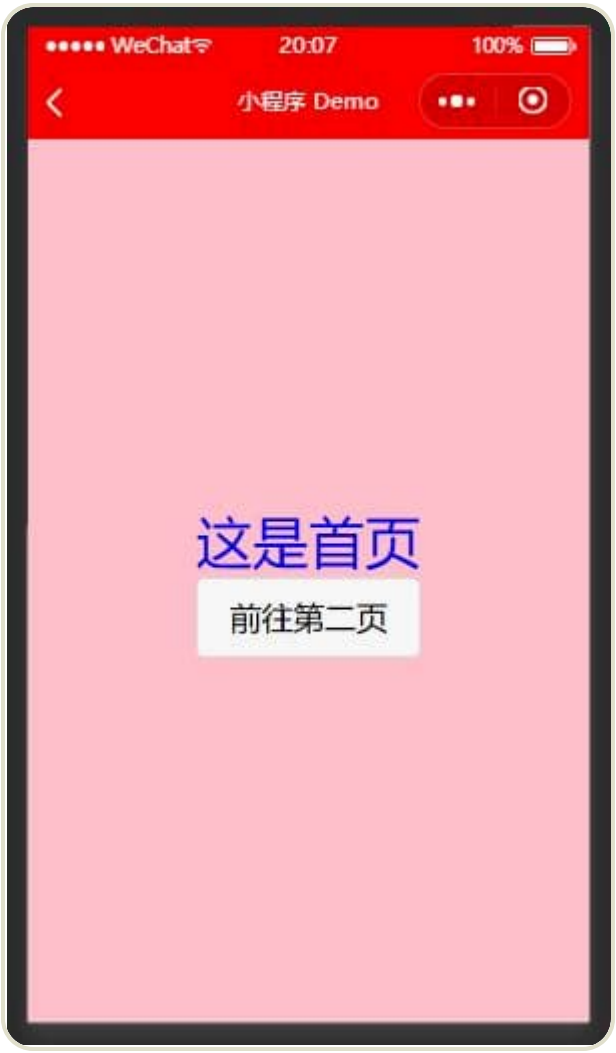
七、wx.navigateTo()

除了使用 `<navigator>` 组件进行页面跳转，小程序也提供了页面跳转的脚本方法 `wx.navigateTo()` 。

首先，打开 `home.wxml` 文件，代码修改如下。

```
<view>
  <text class="title">这是首页</text>
  <button bind:tap="buttonHandler">前往第二页</button>
</view>
```

开发者工具导入项目代码，页面渲染结果如下。



然后，打开 `home.js` 文件，代码修改如下。

```
Page({
  buttonHandler(event) {
    wx.navigateTo({
      url: '../second/second'
    });
  }
});
```

上面代码中， `buttonHandler()` 是按钮点击的监听函数，只要用户点击按钮，就会调用 `wx.navigateTo()` 方法。该方法的参数是一个配置对象，该对象的 `url` 属性指定了跳转目标的位置，自动跳转到那个页面。

这个示例的完整代码，可以参考代码仓库。

写到这里，这个小程序入门教程就告一段落了，入门知识基本上都涉及了。下一步，大家可以阅读小程序的官方教程和使用文档，争取对小程序 API 有一个整体的把

握，然后再去看看各种实际项目的源码，应该就可以动手开发了。以后，我还会写小程序的进阶教程，包括云开发，介绍如何写小程序的后端，下次再见。

（完）

相关文章

- **2020.12.13:** 《SSH 入门教程》发布了

  > SSH 是登录 Linux 服务器的必备工具，只要你在做互联网开发，多多少少都会用到它。

- **2020.10.29:** 微信小程序入门教程之三：脚本编程

  > 这个系列教程的前两篇，介绍了小程序的项目结构和页面样式。

- **2020.10.27:** 微信小程序入门教程之二：页面样式

  > 这个系列的上一篇教程，教大家写了一个最简单的 Hello world 微信小程序。

- **2020.10.26:** 微信小程序入门教程之一：初次上手

> 微信是中国使用量最大的手机 App 之一，日活跃用户超过3亿，月活跃用户超过11亿（2019年底统计），市场极大。

Weibo | Twitter | GitHub

Email: yifeng.ruan@gmail.com