第 5 章

网络与数据 API

前面章节主要介绍了如何编写静态页面，若要将应用真正达到“可用”的程度，还需要通过网络技术来获取服务数据，之后将数据填充到页面上。本章主要对网络请求、文件上传与下载相关的 API 进行讲解。

本章内容：

- 网络数据请求
- 文件上传
- 文件下载
- 微信分享
- 数据缓冲

5.1 网络数据请求

5.1.1 在小程序中调用天气预报 API 服务

网络数据请求实际上是指小程序客户端从服务后台获取数据的能力，此服务后台可以是产品的后端服务，可以是第三方的数据服务，也可以是小程序云开发服务，其实，使用小程序云开发的方式构建后端服务是相对方便且成本较低的，关于云开发后续会进行具体介绍，本节主要讨论从业务服务后台或第三方服务后台获取数据的方法。

小程序通过 wx.request API 向服务器发送 HTTP 请求。该 API 中的常用参数如表 5-1 所示。

表 5-1 wx.request API 中的常用参数

属 性	类 型	说 明
url	String	小程序向开发者服务器访问接口地址
data	String/Object/ArrayBuffer	请求的参数
header	Object	设置请求的 header，header 中不能设置 Referer。 content-type 默认为 application/json
timeout	Number	超时时间，单位为毫秒

续表

属 性	类 型	说 明
method	String	HTTP 请求方法,默认为 GET
dataType	String	返回的数据格式
responseType	String	响应的数据类型
success	函数	接口调用成功的回调函数
fail	函数	接口调用失败的回调函数
complete	函数	接口调用结束的回调函数(调用成功、失败都会执行)

当小程序发送网络请求时,根据情景的不同选择不同的请求方式。HTTP 协议定义了多种请求方法,每种方法都有其特定的用途和特点。以下是一些常见的 HTTP 请求方法及其简要描述。

(1) GET:用于请求服务器发送指定资源。这是最常见的请求方法,通常用于获取网页内容或查询数据。

(2) POST:用于向服务器提交数据,通常会导致服务器状态的改变或产生副作用,例如,用户登录信息或者上传文件通常会使用 POST 方法。

(3) PUT:用于更新服务器上的资源。如果资源不存在,则服务器应该创建它。

(4) DELETE:用于删除服务器上的资源。

(5) PATCH:用于对资源进行部分更新,与 PUT 不同,PUT 通常用于完全替换资源。

其中 GET 和 POST 两种请求方式的使用频率最高,wx.request API 的代码如下:

```
wx.request({
  url: '服务器地址',
  method:'GET'       //请求方式
  data:{
    x:'',
    y''
  },
  header:{
    'content-type':'application/json'
  },
  success(res){
    console.log("请求成功")
  },
  fail(res){
    console.log('请求失败')
  },
  complete(res){
    console.log("请求结束")
  }
})
```

小程序相当于一个 C/S(客户端/服务器)架构应用程序,网络请求的过程是小程序向服务器发送一个 request 请求后,经过服务器处理与查询数据库,将小程序所需的数据以

JSON 格式返回，小程序通过 success()回调函数来接收返回数据，然后经过页面渲染进行数据显示。如果请求失败，则小程序会调用 fail()回调函数，具体的请求过程如图 5-1 所示。

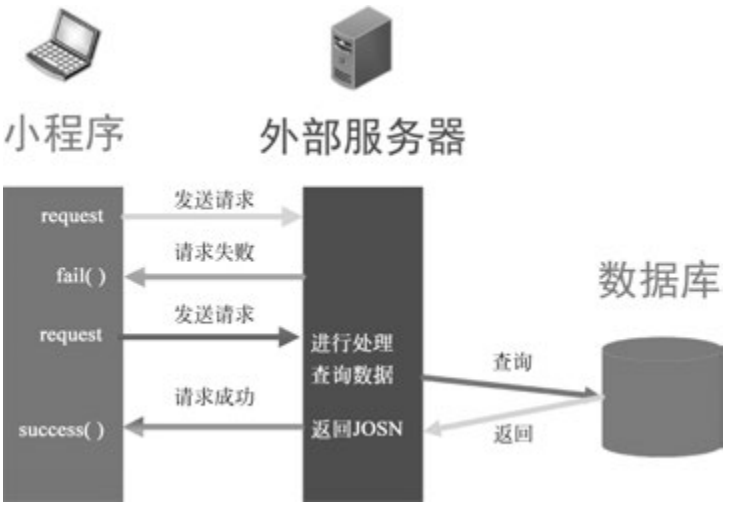


图 5-1　小程序网络请求示意图

为了确保所有小程序在运行时的安全性，微信小程序官方要求所有线上版本的小程序必须使用 HTTPS 协议进行网络通信，不满足条件的域名和协议将无法发出请求(项目要上线时必须设置域名)。在本地开发环境中，可以通过关闭项目设置中的 TLS 安全检测来使用非 HTTPS 地址进行测试。在开发工具的“详情”→“本地设置”中勾选“不校验合法域名”，如图 5-2 所示。

图 5-2　关闭域名校验

使用小程序网络 API 的相关应用制作一款天气查询小程序，通过 wx.request 接口访问一个第三方 API 实现天气查询功能。第三方天气 API 很多，可选用免费版本，如本书使用和风天气官网进行演示。

可自行到和风天气官网申请 API 的密钥，如图 5-3 所示。

图 5-3　申请天气 API 密钥

在示例工程的 pages 文件夹下新建一组名为 requestDemo 的页面文件，在 requestDemo.wxml 文件中编写界面布局，代码如下：

```
<!-- pages/requestDemo/requestDemo.wxml -->
<view class="container">
  <!-- 区域 1:地区选择器 -->
  <picker mode="region" bindchange="regionChange">
    <view>{{region}}</view>
  </picker>
  <!-- 区域 2: 单行天气信息 -->
  <text>{{now.temp}}℃ {{now.text}}</text>
  <!-- 区域 3: 天气图标 -->
  <image src="../images/999.png" mode='widthFix'></image>
  <!-- 区域 4: 多行天气信息 -->
  <view class="detail">
    <view class="bar">
      <view class="box">湿度</view>
      <view class="box">气压</view>
      <view class="box">能见度</view>
    </view>
    <view class="bar">
      <view class="box">{{now.humidity}} %</view>
      <view class="box">{{now.pressure}} hPa</view>
      <view class="box">{{now.vis}} km</view>
    </view>
    <view class="bar">
      <view class="box">风向</view>
      <view class="box">风速</view>
      <view class="box">风力</view>
```

```
      </view>
      <view class="bar">
        <view class="box">{{now.windDir}}</view>
        <view class="box">{{now.windSpeed}} km/h</view>
        <view class="box">{{now.windScale}} 级</view>
      </view>
    </view>
</view>
```

在 requestDemo.wxss 文件中编写样式代码，代码如下：

```
/* pages/requestDemo/requestDemo.wxss */
/*文本样式*/
text {
  font-size: 80rpx;
  color: #3C5F81;
}
/*图标样式*/
image{
  width: 220rpx;
}
/*区域4整体样式*/
.detail{
  width: 100%;
  display:flex;
  flex-direction: column;
}
/*区域4单元行样式*/
.bar{
  display: flex;
  flex-direction: row;
}
/*区域4单元格样式*/
.box{
  width: 33.3%;
  text-align: center;
  height: 30px;
  background-color: white;
}
```

接下来需要在 JS 文件中通过 wx.request 接口访问和风天气提供的第三方 API，获取天气数据，实现天气查询功能，代码如下：

```
//pages/requestDemo/requestDemo.js
//导入数据文件
var util = require('../../utils/request.js')
Page({
  /**
```

```
    * 页面的初始数据
    */
  data: {
    region:['北京市','北京市','东城区'],
    now:{
      temp:0,
      text:'未知',
      humidity:0,
      pressure:0,
      vis:0,
      windDir:0,
      windSpeed:0,
      windScale:0
    }
  },
//更新省、市、区信息
regionChange:function(e){
  this.setData({region:e.detail.value});
  this.getWeather();
},
/*
 * 获取实况天气数据
 */
  getWeather:function(){
    var that = this;                    //this 不可以直接在 wxAPI 函数内部使用
    //获取位置 ID
    var location_name = util.getLocationID(that.data.region[1])
    wx.request({
      //访问和天气提供的 API
      url:'https://devapi.qweather.com/v7/weather/now',
      data:{
        location:location_name,      //城市 ID
        //替换成自己申请的 API 密钥
        key:'f0b56388cf2b47a88178dbaa3d30e2f9'
      },
      success:function(res){
        that.setData({now:res.data.now});
        //console.log(res.data);
      }
    })
  },
  /**
   * 生命周期函数,用于监听页面加载
   */
  onLoad: function (options) {
    this.getWeather();
  },
})
```

request.js 文件(在源码中下载)用于存放国内所有城市地区数据信息和自定义函数 getLocationID(location_name)(用于获取城市 ID),最后使用 module.exports 暴露函数接口,以供页面 js 文件使用。

以上代码的运行效果如图 5-4 所示。

(a) 天气信息显示界面　　(b) 单击切换地址查看天气

图 5-4　天气查询效果

5.1.2　关于 RequestTask 对象

调用 wx.request 方法后,其立即会返回一个 RequestTask 对象,可以使用此对象来对请求过程进行监听或处理,例如可以提前终止请求。

我们知道,通过网络来获取服务数据是有时间代价的,在数据未返回之前,用户可能已经退出了当前页面,此时将无须再等待数据返回,也无须再处理请求完成后的逻辑,对于这种场景,可以提前结束请求。

RequestTask 对象进行 request 请求管理的方法如下。

(1) RequestTask.abort():终端请求任务。

(2) RequestTask.onHeadersReceived(function callback):监听 HTTP Response Header(请求响应)事件会比请求完成事件更早。

(3) RequestTask.offHeadersReceived(function callback):取消监听 HTTP Response Header(请求)事件。

RequestTask 对象应用的代码如下:

```
const RequestTask = wx.request({
    url: '访问地址',
    data:{
      x:'',
      y:''
    },
    header:{
      'content-type':'application/json'
    },
    success(res){
      console.log(res.data)
    }
  });
  RequestTask.abort(); //手动终止请求
```

5.2　文件上传与下载

在微信小程序的开发过程中，文件上传与下载是非常重要的功能之一，特别是在需要用户提交图片、视频或者其他文件的情况下。本节将探讨如何在微信小程序中实现文件上传和下载功能。

5.2.1　文件上传

文件上传是指将用户的本地文件（如用户头像、商品图片、订单附件等）上传至服务器的过程。这在社交应用、电商平台、企业办公应用等方面非常常见。微信小程序提供了 wx.uploadFile 接口来实现这一功能。

将本地资源上传到服务器，首先我们要创建服务器，在小程序开发阶段，一般需要使用本地服务器来对小程序的功能进行开发与测试。本地服务器编程语言一般有 PHP、Java、Python、Node.js 等。适用的本地服务器，选用合适的编程语言与本地服务器可以更好地进行开发。

本书案例使用 Node.js 作为服务器，在项目的根目录下复制文件夹 server（在源码中下载），进入这个文件夹，右击打开外部终端窗口，如图 5-5 所示。

图 5-5　打开外部终端窗口

在终端窗口中安装服务，等待服务安装完毕后，启动服务器，使用如下命名，如图 5-6 所示。

至此将窗口小化即可，服务器正常启动。之后上传文件的接口网址为 http://localhost:3000/api/upload。

图 5-6 安装并启动服务

使用 wx.uploadFile()方法进行上传,此方法可配置的属性如表 5-2 所示。

表 5-2 wx.uploadFile()方法的属性

属性名	类型	说 明
url	String	开发者服务器地址
filePath	String	要上传文件资源的路径(本地路径)
name	String	文件对应的 key,开发者在服务器端可以通过这个 key 获取文件的二进制内容
formData	Object	HTTP 请求中其他额外的表单数据
header	Object	请求头字段
timeout	Number	设置超时时间,单位为毫秒
success	函数	接口调用成功的回调函数
fail	函数	接口调用失败的回调函数
complete	函数	接口调用结束的回调函数(调用成功、失败都会执行)

在示例工程的 pages 文件夹下新建一组名为 uploadDemo 的页面文件,在 uploadDemo.wxml 文件中编写的代码如下:

```
<!-- pages/uploadDemo/uploadDemo.wxml -->
<button type = "primary" bindtap = "bindUploadHandle">上传文件</button>
```

在 uploadDemo.js 文件中编写 chooseImage 和 uploadFile 事件函数代码,实现上传功能,代码如下:

```
//pages/uploadDemo/uploadDemo.js
Page({
  bindUploadHandle() {
      //上传图片,先选择图片
      wx.chooseImage({
          success(res) {
```

```
                //获取用户选中的图片
                const tempFilePaths = res.tempFilePaths
                wx.uploadFile({
                    //服务器地址
                    url: 'http://localhost:3000/api/upload',
                    filePath: tempFilePaths[0],
                    name: 'file',
                    formData:{
                        'user': 'test'
                    },
                    timeout:50000,
                    success(res) {
                        wx.showToast({
                            title:"上传成功"
                        })
                        console.log(res.data);
                    },
                    fail(err){
                        console.log(err);
                    },
                    complete(){
                        console.log("完成");
                    },
                //上传进度监听
                    progressUpdate: function (res) {
                      if (res.progress > 0) {
                        console.log('上传进度: ' + res.progress + '%');
                      }
                })
            }
        })
    }
})
```

运行以上代码，单击“上传文件”按钮，提示上传成功后，可在 server-upload 文件下查看上传成功的文件，如图 5-7 所示。

如果要在手机端测试，则需要注意以下几点：

(1) 手机和计算机必须在同一个网络下。

(2) 要将服务器地址中的 localhost 更换成本机 IP。

5.2.2 文件下载

文件下载是指从服务器获取文件并保存到客户端的过程。微信小程序提供了 wx.downloadFile 接口来实现文件的下载。wx.downloadFile()方法可配置的属性如表 5-3 所示。

图 5-7　文件上传成功

表 5-3　wx.downloadFile()方法的属性

属　性　名	类　　型	说　　明
url	String	所要下载资源的地址
filePath	String	文件下载成功后存储的路径（本地路径）
header	Object	下载文件时发送的网络请求头，header 不能设置 referer
timeout	Number	设置超时时间，单位为毫秒
success	函数	接口调用成功的回调函数
fail	函数	接口调用失败的回调函数
complete	函数	接口调用结束的回调函数（调用成功、失败都会执行）

可以通过网络将图片文件下载到本地并进行渲染，在示例工程的 pages 文件夹下新建一组名为 downloadDemo 的页面文件，在 downloadDemo.wxml 文件中编写的代码如下：

```
<!-- pages/downloadDemo/downloadDemo.wxml -->
<button type = "primary" bindtap = "download">下载文件</button>
<image src = "{{imagePath}}"></image>
```

在 downloadDemo.js 文件中编写的代码如下：

```
//pages/downloadDemo/downloadDemo.js
Page({
    data:{
        imagePath:""
    },
    download:function() {
        let downloadTask = wx.downloadFile({            //下载文件
```

```
            url: 'http://huishao.cc/img/head - img.png',    //文件路径
           //下载完成后保存到的位置
            filePath: wx.env.USER_DATA_PATH + "/1.png",
            success:(res) = >{                              //下载成功的回调
              console.log(res);
              this.setData({
                  imagePath:res.filePath
              });
            },
            fail:(error) = >{                               //下载失败的回调
              console.log(error);
            }
          });
          downloadTask.onProgressUpdate((res) = >{          //监听下载进度
              console.log("资源总长度: " + res.totalBytesExpectedToWrite);
              console.log("已下载: " + res.totalBytesWritten);
              console.log("进度百分比: " + res.progress);
          });
          let fs = wx.getFileSystemManager();               //获取文件管理器
          try {
              fs.accessSync(wx.env.USER_DATA_PATH + "/1.png")
              //检查文件是否存在
              console.log("文件存在");
          } catch (e) {
              console.log("文件不存在");
          }
      }
  })
```

运行以上代码的效果如图 5-8 所示。

图 5-8　下载成功

5.3　微信分享

在微信小程序的开发过程中,集成微信分享和朋友圈分享功能可以极大地提高小程序的曝光度和用户活跃度。

当分享给好友和分享到朋友圈时需要分别添加不同的函数。

- 分享给好友：onShareAppMessage()
- 分享到朋友圈：onShareTimeline()

5.3.1 分享好友

微信分享是指用户将小程序内的某个页面分享给微信好友或微信群组。这通常会包含一些预设的信息，如标题、封面图和描述。

onShareAppMessage 方法中常用的属性如表 5-4 所示。

表 5-4 onShareAppMessage 方法中常用的属性

字 段	默 认 值	说 明
title	当前小程序名称	转发标题
path	当前页面 path，必须是以/开头的完整路径	转发路径
imageUrl	使用默认截图	自定义图片路径，可以是本地文件路径、代码包文件路径或者网络图片路径，支持 PNG 及 JPG，显示图片长宽比是 5∶4

在示例工程的 pages 文件夹下新建一组名为 shareDemo 的页面文件，在 shareDemo.js 文件中编写的代码如下：

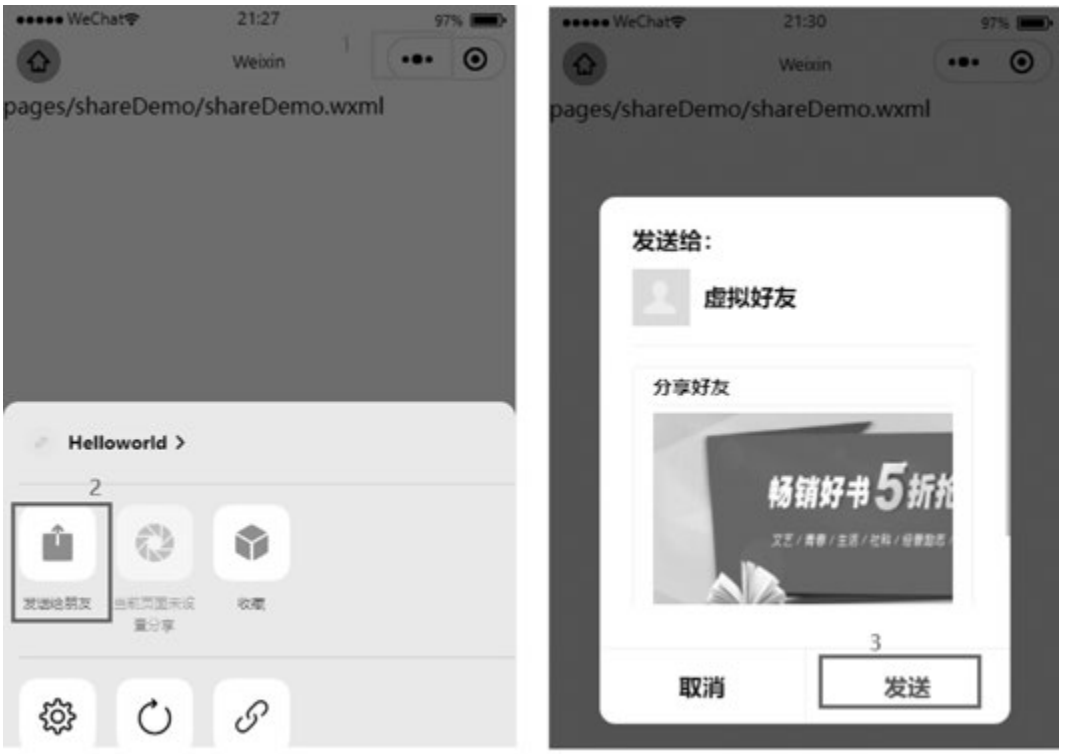
```
//pages/shareDemo/shareDemo.js
Page({
  /**
   * 用户单击右上角的分享
   */
  onShareAppMessage() {
     return {
           title: '分享好友',
           path: '/pages/shareDemo/shareDemo',
           imageUrl:"../images/banner1.jpg",
           menus: ['shareAppMessage']
      }
  }
})
```

单击页面右上角三个点时的效果如图 5-9 所示。

5.3.2 分享朋友圈

朋友圈分享则是指用户将小程序的内容分享到自己的微信朋友圈中。同样地，也会包括标题、封面图和描述等信息。

onShareTimeline 方法中常用的属性如表 5-5 所示。



(a) 单击分享　　(b) 发送好友

图 5-9　分享好友示例

表 5-5　onShareTimeline 方法中常用的属性

字　段	默 认 值	说　明
title	当前小程序名称	自定义标题，即朋友圈列表页上显示的标题
query	当前页面路径携带的参数	自定义页面路径中携带的参数，如 path?a=1&b=2 的“?”后面部分
imageUrl	默认使用小程序 Logo	自定义图片路径，可以是本地文件或者网络图片。支持 PNG 及 JPG，显示图片长宽比是 1∶1

在示例工程 shareDemo 页面文件下的 shareDemo.js 文件中编写的代码如下：

```
//pages/shareDemo/shareDemo.js
Page({
  /**
   * 用户单击右上角分享
   */
  onShareAppMessage() {
     return {
          title: '分享好友',
          path: '/pages/shareDemo/shareDemo',
```

```
            imageUrl:"../images/banner1.jpg",
            menus: ["shareAppMessage", "shareTimeline"],
            //开启“发送给朋友”“分享到朋友圈”
        }
    },
      //用户单击右上角分享到朋友圈
      onShareTimeline(){//点亮分享朋友圈,暂只支持 Android 平台
        return{
          title:'分享到朋友圈',
          query:'/pages/shareDemo/shareDemo',
          imageUrl:"../images/banner1.jpg"
        }
      }
})
```

以上代码的运行效果如图 5-10 所示。

图 5-10　分享到朋友圈示例

5.4　刷新接口

在这个快节奏的时代,用户对信息的渴望如同干渴的沙漠旅人寻找甘泉,而微信小程序的下拉刷新与上拉加载功能,正是那场及时雨,让用户体验如丝般顺滑。

微信小程序可以通过 onPullDownRefresh 和 onReachBottom 两种方法来实现下拉刷新和上拉加载更多功能。

5.4.1　下拉刷新

下拉刷新,顾名思义,用户轻轻下拉页面,即可触发数据的更新,仿佛按下了一个“世界重启”按钮。它让最新内容瞬间呈现,满足用户的实时需求,例如微信的朋友圈。

在小程序页面的 onPullDownRefresh 方法中,可以先调用数据请求函数,然后在请求

成功后调用 wx.stopPullDownRefresh()来停止下拉刷新的动画。

在示例工程的 pages 文件夹下新建一组名为 pullDemo 的页面文件。

首先,在需要添加下拉刷新效果的页面的 pullDemo.wxml 文件中添加下拉刷新组件,用于展示下拉刷新的动画和提示信息,代码如下:

```
<!-- pages/pullDemo/pullDemo.wxml -->
<view class="root">
  <view wx:for="{{list}}" wx:key="index">
    <view class="item">{{item}}</view>
  </view>
</view>
```

在页面的 pullDemo.json 配置文件中开启下拉刷新功能并添加刷新提示信息,代码如下:

```
{
  "usingComponents": {},
  "enablePullDownRefresh": true,
  "backgroundColor": "#f1f1f1",
  "backgroundTextStyle":"dark"
}
```

在页面的 pullDemo.js 文件中,设置下拉刷新相关的数据和事件处理函数,代码如下:

```
//pages/pullDemo/pullDemo.js
Page({
  /**
   * 页面的初始数据
   */
  data: {
    list:[1,2,3,4,5]
  },
  /**
   * 页面相关事件处理函数,用于监听用户下拉动作
   */
  onPullDownRefresh() {
   setTimeout(()=>{
   //下拉后切换的数据,可以请求网络数据
      this.setData({
        list:[6,7,8,9,10]
      })
      //请求成功后停止下拉刷新动画
      wx.stopPullDownRefresh();
    },1000)
  }
})
```

在页面的 pullDemo.wxss 文件中添加样式，代码如下：

```
/* pages/pullDemo/pullDemo.wxss */
page{
  background: #fff;
}
.root{
  padding: 10px;
}
.item{
  width: 100%;
  height: 50px;
  border-bottom: 1px solid #afafaf;
  line-height: 50px;
}
```

通过上述步骤的操作，已经实现了微信小程序中下拉刷新效果。用户在页面中下拉即可触发刷新，并通过 onPullDownRefresh 事件监听刷新动作，实现数据的及时更新。

需要注意的是，onPullDownRefresh 事件只能在具有下拉刷新样式的页面中生效，如果没有设置页面的 backgroundColor、backgroundTextStyle 和 navigationBarBackgroundColor，则下拉刷新会无效。另外，当刷新完成后，需要调用 wx.stopPullDownRefresh()函数来停止下拉刷新，否则页面将保持刷新状态。

5.4.2 上拉加载

上拉加载，则用于鼓励用户探索更多未知的“魔法”。当用户浏览至页面底部时，自动或手动上拉即可加载更多内容，宛如打开了一扇新世界的大门。

在小程序页面的 onReachBottom 方法中，首先可以调用加载更多数据的函数，然后在请求成功后将新数据追加到已有数据列表中。

在示例工程的 pages 文件夹下新建一组名为 bottomDemo 的页面文件。

首先，在需要添加上拉加载效果页面的 bottomDemo.wxml 文件中添加上拉加载组件，代码如下：

```
<!-- pages/bottomDemo/bottomDemo.wxml -->
<view class="root">
    <view class="item" wx:for="{{ list }}" wx:key="index">
        <text>{{ item }}</text>
    </view>
</view>
```

上拉触底距离指的是触发上拉触底事件时，滚动条距离页面底部的距离。

可以在全局或者页面的 bottomDemo.json 文件中通过 onReachBottomDistance 属性配置上拉触底的距离，代码如下：

```
{
  "usingComponents": {},
  "onReachBottomDistance": 50
}
```

在页面的 bottomDemo.js 文件中，设置上拉加载相关的数据和事件处理函数，示例代码如下：

```
//pages/bottomDemo/bottomDemo.js
Page({
  data: {
      list:[1,2,3,4,5]
  },
  /**
   * 页面上拉触底事件的处理函数
   */
  onReachBottom() {
      this.setData({
          list:this.data.list.concat([6,7,8,9,10])
      })
  }

})
```

在页面的 bottomDemo.wxss 文件中添加样式，代码如下：

```
/* pages/bottomDemo/bottomDemo.wxss */
.item{
  height: 200px;
}
text{
  font-size: 30px;
}
```

通过以上步骤，就可以在微信小程序中实现上拉加载功能。

5.4.3 返回顶部

在页面中添加一个返回顶部的按钮，并绑定单击事件，代码如下：

```
<!-- pages/bottomDemo/bottomDemo.wxml -->
<button bindtap="goToTop" style="position: fixed; bottom: 100px; right: 20px;">返回顶部
</button>
```

在.js 文件中实现 goToTop 方法，代码如下：

```
//pages/bottomDemo/bottomDemo.js
Page({
  goToTop: function() {
    wx.pageScrollTo({
      scrollTop: 0,
      duration: 300 //回滚动画时间,单位为毫秒
    });
  }
});
```

5.5 数据缓冲

微信小程序是一种轻量级的应用程序,小程序开发中的数据缓存可以提高小程序的性能和用户体验,数据缓存可以将数据暂时存储在客户端的内存中,以减少网络请求的次数,提高数据的加载速度。

微信小程序提供了两种缓冲方式。

(1) 通过 wx.setStorageSync 方法进行同步缓冲的设置,适用于缓存小量数据。

(2) 通过 wx.setStorage 方法进行异步缓冲的设置,适用于缓存较大量数据。

5.5.1 同步缓冲

微信小程序提供了一个同步缓存的 API,用于将数据存储在客户端的本地存储空间中。同步缓存是基于键-值对的存储方式,可以使用以下 API 进行操作。

1. wx.setStorageSync(key, data)

该方法用于将数据存储到同步本地缓存中,其中 key 是键,data 是值。可以存储的数据类型包括字符串、数字、对象、数组等,代码如下:

```
wx.setStorageSync('name', 'Tom');
```

2. wx.getStorageSync(key)

该方法用于从同步本地缓存中读取数据,其中 key 是要读取的数据的键,代码如下:

```
var name = wx.getStorageSync('name');
console.log(name); //输出: Tom
```

3. wx.removeStorageSync(key)

该方法用于从同步本地缓存中移除指定的数据,其中 key 是要移除的数据的键,代码如下:

```
wx.removeStorageSync('name');
```

4. wx.clearStorageSync()

该方法不需要传入任何参数,它会清除所有本地数据缓存。

在示例工程的 pages 文件夹下新建一组名为 storageSyncDemo 的页面文件。

在 storageSyncDemo.js 文件中填写的代码如下：

```
//pages/storageSyncDemo/storageSyncDemo.js
Page({
   onLoad(options) {
            wx.setStorageSync("color","red")
            var value = wx.getStorageSync('color');
            console.log(value);
    // wx.removeStorageSync('color')
    // wx.clearStorageSync()
      }
})
```

温馨提示：异步不会阻塞当前任务，同步缓存直到同步方法处理完才能继续往下执行。通俗地讲：异步就是不管是否保存成功，程序都会继续往下执行。同步是等保存成功了，才会执行下面的代码。

使用异步，性能会更好，而使用同步，数据会更安全。

5.5.2 异步缓冲

除了同步数据缓存，微信小程序还提供了一个异步本地存储的 API，用于将数据存储在客户端的本地存储空间中，可以使用以下 API 进行操作。

1. wx.setStorage(options)

该方法用于将数据存储在本地缓存中指定的 key 中。会覆盖原来该 key 对应的内容。除非用户主动删除或因存储空间原因被系统清理，否则数据都一直可用。单个 key 允许存储的最大数据长度为 1MB，所有数据存储上限为 10MB。该接口中常用的参数如表 5-6 所示。

表 5-6 wx.setStorage(options)中的常用参数

属性	类型	必填	说 明
key	String	是	本地缓存中指定的 key
data	Any	是	需要存储的内容。只支持原生类型、Date 及能够通过 JSON.stringify 序列化的对象
success	函数	否	接口调用成功的回调函数
fail	函数	否	接口调用失败的回调函数
complete	函数	否	接口调用结束的回调函数(调用失败、成功都会执行)
encrypt	Boolean	否	是否开启加密存储。只有异步的 setStorage 接口支持开启加密存储。开启后，将会对 data 使用 AES128 加密，接口回调耗时将会增加。若开启加密存储，setStorage 和 getStorage 需要同时将 encrypt 的值声明为 true。此外，由于加密后的数据会比原始数据膨胀 1.4 倍，因此开启 encrypt 的情况下，单个 key 允许存储的最大数据长度为 0.7MB，所有数据存储上限为 7.1MB

温馨提示：AES 加密为高级加密标准(Advanced Encryption Standard，AES)，是一种区块加密标准。AES 可以使用 128、192 和 256 位密钥，从安全性来看，AES256 安全性最高。从性能来看，AES128 性能最高。

wx.setStorage()接口的代码如下：

```
//pages/storageSyncDemo/storageSyncDemo.js
Page({
      onLoad(options) {
          wx.setStorage({
              key: "username",
              data: "admin",
              encrypt: true
          })
      }
 })
```

运行以上代码的效果如图 5-11 所示。

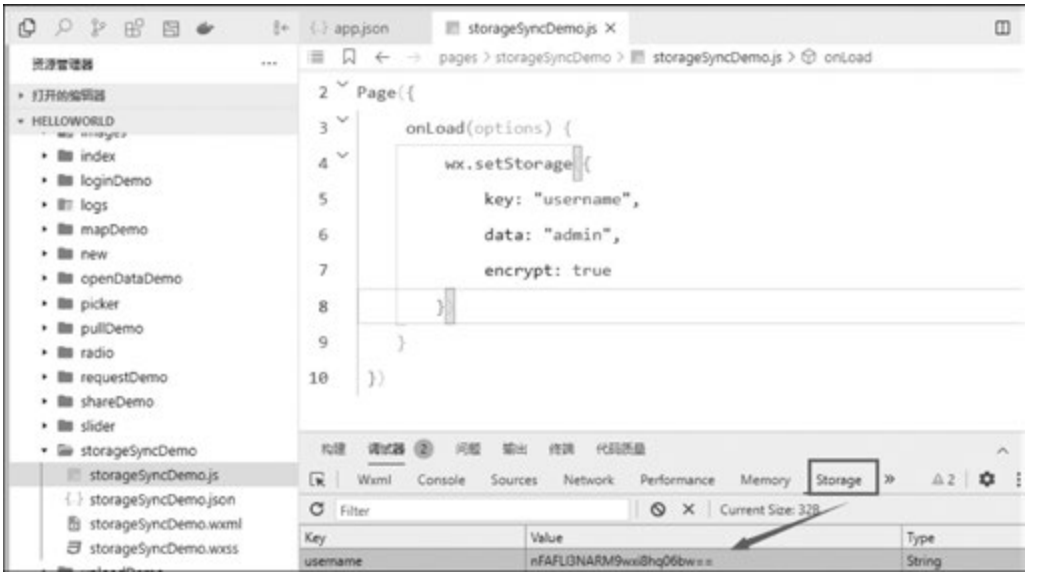


图 5-11　异步缓冲效果

2. wx.getStorage(options)

该方法用于从本地存储中读取数据，代码如下：

```
//pages/storageSyncDemo/storageSyncDemo.js
Page({
      onLoad(options) {
          wx.setStorage({
              key: "username",
```

```
            data: "admin",
            encrypt: true
          })
        wx.getStorage({
          key: 'username',
          success: function (res) {
            console.log(res.data);
          },
          fail: function (res) {
            console.log('读取失败');
          }
        })
    }
})
```

3. wx.removeStorage(options)

该方法用于从本地存储中移除指定的数据,代码如下:

```
//pages/storageSyncDemo/storageSyncDemo.js
Page({
    onLoad(options) {
        wx.setStorage({
            key: "username",
            data: "admin",
            encrypt: true
        })
        wx.removeStorage({
              key: 'username',
              success(res) {
                  console.log(res)
              }
        })
    }
})
```

为了避免意外,最好用 try-catch 语句进行捕获,代码如下:

```
//pages/storageSyncDemo/storageSyncDemo.js
Page({
    onLoad(options) {
        wx.setStorage({
            key: "username",
            data: "admin",
            encrypt: true
        })
        try {
            wx.removeStorage({
```

```
                    key: 'username',
                    success(res) {
                        console.log(res)
                    }
                })
            } catch (e) {
                //发生意外
                console.log(e);
            }
        }
})
```

4. wx.clearStorage()

清理本地数据缓存,代码如下:

```
Page({
    onLoad(options) {
        wx.setStorage({
            key: "username",
            data: "admin"
        })
        wx.clearStorage()
    }
})
```