第 5 章

JavaScript 基础

JavaScript 是小程序逻辑层的编程语言，JavaScript 代码大约占小程序代码量的一半，学好 JavaScript 是学好小程序开发的关键。

本章主要目标

- 熟练掌握 JavaScript 语法格式；
- 熟练掌握 JavaScript 变量、数据类型、运算符、函数等基本概念；
- 熟练掌握小程序事件函数中 this 和 that 的使用；
- 熟练掌握 JavaScript 在小程序中的交互场景应用。

5.1 JavaScript 简介

JavaScript 是一种轻量、解释型、支持面向对象编程风格的脚本语言，它是一种直译式、动态类型、弱类型以及基于原型的语言。JavaScript 语言不仅可用于 Web 前端开发，也广泛用于后端开发和智能手机开发。JavaScript 的标准是 ECMAScript。2015 年 6 月 17 日，ECMA 国际组织发布了 ECMAScript 的第六版，该版本正式名称为 ECMAScript 2015，但通常被称为 ECMAScript 6 或者 ES6。

JavaScript 具有以下特性：

- JavaScript 是一种基于对象的脚本语言，它不仅能够创建对象，而且可以使用对象；
- JavaScript 是一种轻量级的编程语言；
- JavaScript 是可插入 HTML 页面的编程代码；
- JavaScript 插入 HTML 页面后，可由现在所有的浏览器执行。

正是由于 JavaScript 易学易用的特性，使得它在最近几年应用广泛，获得了编程界的好评，同时出现了大量基于 JavaScript 的开源项目。如今 JavaScript 不仅可以实现前端的动态交互功能，而且可以运行于后端，为用户提供高性能的后端服务。在微信小程序中，

JavaScript 是小程序逻辑层使用的唯一开发语言，这也客观反映出 JavaScript 的强大，因此学好 JavaScript 对实现前端的交互功能非常重要。

在 iOS 系统中，小程序的 JavaScript 代码是运行在 JavaScriptCore 中，然后由 WKWebView 来渲染。

在安卓系统中，小程序的 JavaScript 代码是由 X5 JS core 来解析，由 X5 内核渲染。

在微信开发者工具(IDE)中，小程序的 JavaScript 代码是运行在 nw.js 中，由 Chrome WebView 来渲染。其中，nw.js 是基于 Chromium 和 Node.js 运行的，封装了 Webkit 内核和 Node.js，提供了桌面应用的运行环境，让在浏览器运行的程序也可以在桌面端运行。

这 3 个运行环境(iOS、安卓和微信开发者工具)使用的 ECMA 标准是不一样的，目前 ECMAScript(简称 ES)有 8 个版本，小程序使用的是 ES5 和 ES6 标准。但截至目前，iOS 8 和 iOS 9 并没有完全兼容到 ES6 的标准，即 ES6 中的一些语法和关键字不被兼容，所以经常会发现，在微信开发者工具里和手机真机上的代码表现不一致，对此可以用微信开发者工具里的远程调试功能，在真机上进行调试。

5.2 JavaScript 基础语法

本节将介绍 JavaScript 的基础语法，包括变量、数据类型、运算符、逻辑控制语句等。

5.2.1 变量

变量是存储信息的容器，所有 JavaScript 变量必须以唯一的名称标识，标识称为变量名。定义变量名称的规则为：名称可包含字母、数字、下画线；名称必须以字母开头，对大小写敏感(x 和 X 是不同的变量)；JavaScript 的关键词不能作为变量名称。

示例代码：

```
var a = 1;                          //声明变量 a 并且赋值为数字 1
var b = "abc";                      //声明变量 a 并且赋值为字符串 abc
```

如上述代码所示，注释是指编程中用于解释说明的部分，用于提高代码的可读性。JavaScript 支持单行注释和多行注释。单行注释用//标注；多行注释用/*…*/标注。

5.2.2 数据类型

JavaScript 变量能够保存多种数据类型：字符串、数字、逻辑值、数组、对象。本节将逐一介绍。

1. 字符串(String)类型

字符串用于存储一系列字符，使用单引号或双引号包裹字符串的内容。

示例代码：

```
var hello = "Hello xiaochengxu";
```

```
var hello = 'Hello xiaochengxu';
```

无论是单引号还是双引号，都需要成对出现，不能出现不一致现象，否则在编译时会报错。

2. 数字(Number)类型

JavaScript 只有一种数字类型，数值后面的小数点可省略，用科学记数法能够表示极大值和极小值。

```
var x1 = 1.00;                    //带小数点
var x2 = 1;                       //不带小数点
var m = 123e5;                    //12300000
var n = 123e-5;                   //0.00123
```

3. 布尔(Boolean)类型

布尔(逻辑)类型只有两个值：true 和 false，在使用布尔值时，不能出现"false"或"true"，否则会被解析为字符串。

示例代码：

```
var x = true;
var y = false;
```

变量 x 和变量 y 都是布尔类型，如果按照下面写法：

```
var x = "true";
var y = "false";
```

变量 x 和变量 y 都是字符串类型，字符串里面的内容分别为 true 和 false。

4. 数组类型

JavaScript 中的数组用方括号书写，数组中的元素由逗号分隔。

示例代码：

```
var list = ["a", "b", "c", "d", "e", "f"];
```

上述代码定义一个数组，数组名为 list，数组中包含 6 个元素。数组的索引 index 从 0 到数组的个数减 1，通过索引值可以取得数组中的元素。例如 list[2]可以取得数组中的第 3 个元素。数组中包含一些常用的方法，在微信小程序中常用的 Array 对象方法如表 5.1 所示。

表 5.1 Array 对象方法

方 法 名	说 明
pop()	删除并返回数组的最后一个元素
push()	向数组的末尾添加一个或更多元素，并返回新的长度
reverse()	颠倒数组中元素的顺序
shift()	删除并返回数组的第一个元素
slice()	从某个已有的数组返回选定的元素
sort()	对数组的元素进行排序
splice()	删除元素，并向数组添加新元素
toString()	把数组转换为字符串，并返回结果
unshift()	向数组的开头添加一个或更多元素，并返回新的长度

以表 5.1 中的 push()方法为例，示例代码如下：

```
var fruits = ["Apple", "Banana", "Lemon", "Grape"];
fruits.push("Cherry");
for (var i = 1; i < fruits.length; i++) {
  console.log(fruits[i]);
}
```

上述代码执行完以后，会在 Console 控制台输出“Apple，Banana，Lemon，Grape，Cherry”。

5. 对象类型

JavaScript 对象用大括号来书写，内容放置在大括号中，对象的属性通过名称和值(name：value)来定义，属性之间用逗号分隔。示例代码如下：

```
var person = {
  firstName: "tom",
  age: 23,
  hairColor: "black"
};
```

上述代码中对象(person)有 3 个属性：firstName、age 和 hairColor。

5.2.3 运算符

JavaScript 运算符常用于执行算术运算、赋值、比较运算、逻辑运算。

1. 算术运算符

算术运算符用于变量或值之间的算术运算，如表 5.2 所示。

表 5.2 JavaScript 算术运算符

运 算 符	描 述	例 子
+	加	x=y+1
−	减	x=y−1
*	乘	x=y*1
/	除	x=y/1
%	求余数(保留整数)	x=y%1
++	累加	x=++y
−−	递减	x=−−y

对于两个数字型的变量，变量之间使用“+”运算符表示相加；当两个变量都为字符型，“+”运算符表示字符串的连接；当一个变量为字符型，另一个变量为数值型，数值型会被解析为字符型进行字符串的连接。

2. 赋值运算符

对变量进行赋值使用赋值运算符，如表 5.3 所示。

表 5.3 JavaScript 赋值运算符

运 算 符	例 子	等 同 于
=	x=y	
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

3. 比较运算符

比较运算符表示变量之间的逻辑关系，如表 5.4 所示。

表 5.4 JavaScript 比较运算符

运 算 符	描 述	比较及结果
==	等于	5=6 为 false
===	全等(值和类型)	5===5 为 true，5===''5''为 false
=!	不等于	5!=6 为 true
>	大于	5>6 为 false
<	小于	5<6 为 true
>=	大于或等于	5>==6 为 false
<=	小于或等于	5<==6 为 true

4. 逻辑运算符

逻辑运算符用于确定变量或值之间的逻辑关系，假设 x=5 and y=2，逻辑运算符的使用如表 5.5 所示。

表 5.5 JavaScript 逻辑运算符

运 算 符	描 述	例 子
&&	和	(x < 10 && y > 1)为 true
||	或	(x == 3 || y == 5)为 false
!	非	!(x == y)为 true

5.2.4 逻辑控制语句

逻辑控制语句分为条件判断语句和循环语句。

1. 条件判断语句

条件判断语句基于分支的思想，面对不同的情况或条件执行相应的选择，通用于某些代码的判断和重复执行。

1) if 语句

当小括号内的条件为 true 时，大括号内部的代码才会执行。语法如下：

```
if (条件) {
```

```
  当条件为 true 时才执行该代码
}
```

示例代码：

```
if (data > 100) {
  console.log(data)
}
```

上述代码中当变量 data 值大于 100 时，会在 Console 控制台输出这个值。

2）if…else 语句

当条件为 true 时执行 if 之后的代码，当条件为 false 时执行 else 之后的代码。语法如下：

```
if (条件) {
  当条件为 true 时执行该代码
} else {
  当条件为 false 时执行该代码
}
```

示例代码：

```
if (data > 100) {
  console.log("输入的数字大于 100")
} else {
  console.log("输入的数字小于或等于 100")
}
```

上述代码中当变量 data 值大于 100 时，会在 Console 控制台输出“输入的数字大于 100”，否则，会在 Console 控制台输出“输入的数字小于或等于 100”。

3）if…else if…else 语句

在多个代码块中选择满足条件的一个去执行，先判断 if 条件，如果条件不成立，返回 false 后判断 else if 条件，如果条件成立，返回 true 并执行该条 else if 后的代码，如果仍不成立，继续判断下一个 else if 的条件。如果直到 else 时都没有条件成立，则执行 else 中的代码。语法如下：

```
if (条件 1) {
  当条件 1 为 true 时执行该代码
} else if (条件 2) {
  当条件 2 为 true 时执行该代码
} else {
  当条件 1 和条件 2 都为 false 时执行该代码
}
```

示例代码：

```
if (data < 60) {
  console.log("输入的数字小于 60")
} else if (data >= 60 && data <= 80) {
  console.log("输入的数字在 60～80 中")
```

```
} else {
  console.log("输入的数字大于 80")
}
```

当变量 data 值小于 60 时，执行 if 之后的代码，会在 Console 控制台输出“输入的数字小于 60”；当变量 data 值为 60～80 时，执行 else if 之后的代码，输出“输入的数字在 60～80 中”；当前面的条件都不满足时，执行 else 后面的语句，输出“输入的数字大于 80”。

4）switch 语句

在多个代码块中选择满足条件的一个去执行，判断表达式通常为变量，表达式的值会与 case 中的值做匹配，如果匹配正确，会执行该 case 之后的代码块，使用 break 可以阻止代码继续执行，当条件都不满足时执行 default 之后的代码。

示例代码：

```
switch (n) {
  case 1:
    console.log("今天是星期一")
    break;
  case 2:
    console.log("今天是星期二")
    break;
  case 3:
    console.log("今天是星期三")
    break;
  case 4:
    console.log("今天是星期四")
    break;
  case 5:
    console.log("今天是星期五")
    break;
  case 6:
    console.log("今天是星期六")
    break;
  case 7:
    console.log("今天是星期日")
    break;
  default:
    console.log("输入有误请重新输入")
}
```

当判断条件符合其中某一条件时，在 Console 控制台输入对应的星期数，然后程序执行 break 语句跳出循环；当条件不满足所有 case 时，会执行 default 之后的代码，输出“输入有误请重新输入”。

2. 循环语句

循环语句通过判断条件来控制循环的次数，如果需要重复执行相关的动作就需要使用循环语句，循环语句可以提高代码的精简性。

例如，需要输出一个数组的全部元素，不使用循环语句代码如下：

```
console.log(array[0]);
```

```
console.log(array[1]);
console.log(array[2]);
console.log(array[3]);
```

使用循环语句代码如下：

```
for (var i = 0; i < array.length; i++) {
  console.log(array[i]);
}
```

上述代码通过循环将数组 array 中的所有元素在 Console 控制台输出。

1）for 循环

for 循环是经常使用的循环语句，语法如下：

```
for (语句 1; 语句 2; 语句 3)
{
  被执行的代码
}
```

语句 1：在循环开始前执行初始化操作；语句 2：循环的条件，当返回值为 true 时，执行循环体，当返回值为 false 时，跳出该循环；语句 3：当语句 2 返回值为 true 时执行。

示例代码：

```
for (i = 1; i < 10; i++) {
  console.log(i)
}
console.log(i)
```

上述代码会在 Console 控制台分别输出 1 到 9。

2）for/in 循环

for/in 循环用于循环对象属性，当循环体中的代码每执行一次，就会对数组的元素或者对象的属性进行一次操作。

示例代码：

```
var person = {
  firstName: "tom",
  age: 23,
  hairColor: "black"
};
var string = "";
var x;
for (x in person) {
  string += person[x];
}
```

输出结果为：tom 23 black。

3）while 循环

当条件满足时执行代码块，在每次执行完后会进行条件判断，条件为真会再次执行，直到条件不为真时跳出循环。语法如下：

```
while (条件) {
  执行其中的代码
}
```

示例代码：

```
var i = 1;
while (i < 10) {
  console.log("这是代码执行的第" + i + "次");
  i++
}
```

变量 i 从初始值 1，到不满足循环条件 i<10，共执行了 9 次循环，程序在 Console 控制台执行了 9 次语句的输出。

4）do…while 循环

首先执行一次 do 里面的代码块，如果条件为真会重复执行，当条件为假时跳出循环。语法如下：

```
do {
  执行其中的代码
}
while (条件);
```

示例代码：

```
var i = 1;
do {
  console.log("这是代码执行的第" + i + "次");
}
i++;
while (i < 1);
```

首先执行 do 里面的代码，然后变量 i 变为 2，不满足 while 里面的条件直接跳出循环。

5.2.5 定义和调用函数

函数的定义使用的关键字是 function，示例代码：

```
function functionName(parameters) {
  执行的代码
}
```

在小程序中函数的定义如下：

```
functionName:function(e){
  执行的代码
}
```

函数的调用是指使用事先定义好的函数，函数本身是一种对象，示例代码：

```
function myFunction(a, b) {
```

```
  return a + b;
}
myFunction(1, 2);                    //myFunction(1, 2) 返回值为 3
```

5.2.6 小程序中 this 和 that 的使用

this 是 JavaScript 语言的一个关键字，可以调用函数。当函数运行时，this 可以在函数内部使用，当函数使用场合发生变化时，this 的值也会发生变化。在小程序开发中，小程序提供的 API 接口经常会有 success、fail 等回调函数来处理后续逻辑。当需要获取当前页面对象来对视图层进行渲染时，this 只会指向调用函数的对象，如果想要获取页面的初始数据，在回调函数里面就不能使用 this.data 来获取，同时也不能使用 this.setData()函数来更新数据，而是通过语句 var this＝that 将 this 指向的对象复制到 that 中才可以执行后续操作。

5.3 JavaScript 在小程序中常见的交互场景

JavaScript 是小程序编程中的基础语言，从本书附带案例的代码来看，JavaScript 代码大约占整个小程序项目一半的代码量。全局文件 app.js 和所有的页面的 JS 文件都是由 JavaScript 来编写的，JavaScript 代码主要实现业务逻辑处理和用户交互两方面的作用。5.2 节主要从语法的角度介绍了 JavaScript 的变量、数据类型、控制语句和函数定义与调用，本节将以 JavaScript 在小程序中常见的交互场景出发，以案例教学的方式带领读者更深层次地理解 JavaScript 在小程序编程中的使用。

5.3.1 购物车场景

尽管张小龙在 2018 微信公开课上指出，“小程序不是专门为电商准备的”，但由于强社交属性和微信支付的便捷性，电商成为小程序的重要应用场景。

例 5-1 电商小程序中经常要用到购物车，购物车是 JavaScript 在小程序交互场景中的经典应用。本例实现一个简单的购物车，购物车初始状态和用户加购商品之后的购物车状态如图 5.1 和图 5.2 所示。

视频讲解

pages/addCaricon/addCaricon.wxml 文件代码如下：

```
<view class = "title">1.显示图标小案例</view>
<view class = "goods - box">
  <! -- 商品展示 -- >
  <image src = "/images/goods.png"></image>
  <view class = "carts - icon">
    <! -- 购物车图标 -- >
    <image src = "/images/cart.png"></image>
    <! -- 加入购物车的数量 -- >
```

```
    <text class="carts-num" wx:if="{{hasCart}}">{{totalNum}}</text>
  </view>
  <!--加入购物车导航-->
  <view class="operation">
    <text class="operation-num">数量 {{num}}</text>
    <text class="operation-add" bindtap="addCount">+</text>
    <text bindtap="addCart">加入购物车</text>
  </view>
</view>
```

图 5.1 购物车初始状态

图 5.2 加购之后的购物车

pages/addCaricon/addCaricon.js 文件代码如下：

```
Page({
  data: {
    num: 1,                                  //数量初始化为 1
    totalNum: 0,                             //加入购物车的商品数量初始化为 0
    hasCart: false,                          //初始化默认不加入购物车
  },
  //增加数量
  addCount: function() {
    var num = this.data.num;
    num++;                                   //数量增加 1
    this.setData({
      num: num                               //更新数量
```

```
    })
  },
  //加入购物车
  addCart: function() {
    const num = this.data.num;                    //获取数量
    var total = this.data.totalNum;               //获取总数
    this.setData({
      hasCart: true,                              //用于控制加入购物车时的显示图标
      totalNum: num + total                       //累计数量
    })
    wx.showToast({
      title: "加入购物车成功",                     //弹出消息提示框
      duration: 3000,
    })
  },
})
```

pages/addCaricon/addCaricon.wxss 文件代码如下：

```
/*页面盒子*/
.goods-box {
  position: relative;
  padding: 0 30rpx;
  text-align: center;
}
/*购物车位置*/
.carts-icon {
  position: absolute;
  right: 600rpx;
  top: 490rpx;
  width: 70rpx;
  height: 70rpx;
}
/*购物车图标*/
.carts-icon image {
  width: 100%;
  height: 100%;
}
/*显示加入购物车图标*/
.carts-num {
  position: absolute; right: 36rpx; width: 40rpx;
  height: 40rpx; color:white;
  font-size: 26rpx;
  line-height: 40rpx;
  border-radius: 50%;
  background: #e4393c;
}
/*加入购物车导航*/
.operation {
  height: 80rpx; line-height: 80rpx;
```

```
  color:white; border-radius: 50rpx;
  background: #ff9600;
  font-size: 30rpx;
}
/*操作文本*/
.operation text {
  display: inline-block;
  height: 80rpx;
}
/*加入购物车数量*/
.operation-num {
  width: 160rpx;
}
/*加入购物车符号*/
.operation-add {
  width: 80rpx;
  margin-right: 50rpx;
}
```

【代码讲解】 addCaricon.wxml 文件中“+”和“加入购物车”两个按钮绑定了点击事件。在 addcart.js 文件中为“+”按钮定义了事件函数 addCount(),用于实现当用户点击“+”按钮时商品数量加 1。为“加入购物车”按钮定义的函数 addToCart(),用于实现当用户点击“加入购物车”时,一次性向购物车添加 num 件商品。当用户有加购行为,即点击了“加入购物车”按钮时,hasCart 被赋值为 true,则在购物车图标的左上角会出现当前购物车商品数量。

5.3.2 下拉菜单场景

JavaScript 在 Web 开发中经常被用来实现动态效果,在微信小程序开发者中也存在类似的场景。下拉菜单是 JavaScript 在小程序中常见的交互场景之一,当用户单击一级菜单时,二级菜单被弹出,当再次单击一级菜单时,二级菜单又消失。

例 5-2 本例设计常见的小程序下拉菜单,图 5.3 是二级菜单没有弹出的状态,图 5.4 是二级菜单弹出的状态。

pages/subMenu/subMenu.wxml 文件代码如下:

视频讲解

```
<view class="title">2.下拉菜单场景小案例</view>
<view class="page">
  <!--导航内容-->
  <view class="nav">
    <view class="nav-item {{shownavindex == 1?'active':''}}"
      bindtap="listmenu" data-nav="1">
      <view class="content">排序</view>
      <view class="icon"></view>
    </view>
    <view class="nav-item">
      <view class="content">时间</view>
      <view class="icon"></view>
```

```
    </view>
    <view class="nav-item">
      <view class="content">价格</view>
      <view class="icon"></view>
    </view>
  </view>
  <!--下拉内容-->
  <view class="list {{openif?'down':'up'}} ">
    <view wx:for="{{content}}">
      {{item}}
    </view>
  </view>
</view>
```

图 5.3　下拉菜单弹出之前

图 5.4　下拉菜单弹出之后

pages/subMenu/subMenu.js 文件代码如下：

```
Page({
  data: {
    li: ['默认排序', '离我最近', '价格最低', '价格最高'],
    shownavindex: 0                                    //数据初始化
  },
  //下拉事件
  listmenu: function(e) {
    if (this.data.openif) {
      this.setData({
```

```
        openif: false,                                  //当前菜单没有下拉
        shownavindex: 0                                 //控制图标样式
      })
    } else {
      this.setData({
        content: this.data.li,                          //获取数组数据
        openif: true,                                   //当前菜单下拉
        shownavindex: e.currentTarget.dataset.nav       //控制图标样式
      })
    }
  }
})
```

pages/subMenu/subMenu.wxss 文件代码如下：

```
/*页面溢出隐藏*/
.page {
  overflow: hidden;
}
/*导航外部样式*/
.nav {
  position: relative;
  z-index: 1;
  display: flex;
  flex-direction: row;
  background: white;
}
/*导航内部样式*/
.nav-item {
  display: flex;
  flex: 1;
  text-align: center;
  height: 90rpx;
  align-items: center;
  justify-content: center;
  font-size: 30rpx;
  border: 1px solid gray;
}
/*导航下拉图标*/
.icon {
  border: 10rpx solid transparent;
  border-top: 10rpx solid gray;
  margin-left: 12rpx;
}
/*下拉内部样式*/
.list {
  display: none;
  width: 100%;
  overflow-y: scroll;
  padding: 0 0 0 20rpx;
```

```
  line-height: 100rpx;
  background: white;
}
/* 下拉内容样式 */
.list view {
  border-bottom: 1px solid gray;
  font-size: 32rpx;
}
/* 点击导航内容文字后的样式 */
.nav-item.active .content {
  color: skyblue;
}
/* 点击导航下拉图标后的样式 */
.nav-item.active .icon {
  border-bottom: 10rpx solid skyblue;
  border-top: 0;
}
/* 下拉动画样式 */
.down {
  display: block;
  animation: slidown 0.5s ease-in both;
}
@keyframes slidown {
  from {
    transform: translateY(-100%);
  }
  to {
    transform: translateY(0%);
  }
}
/* 收起动画样式 */
.up {
  display: block;
  animation: slidup 0.5s ease-in both;
}
@keyframes slidup {
  from {
    transform: translateY(0%);
  }
  to {
    transform: translateY(-100%);
  }
}
```

【代码讲解】 本例是 JavaScript 和样式布局结合的案例，在 subMenu.js 文件中设置初始值 shownavindex：0，使得导航的字体初始值为黑色。当用户点击菜单的时候激活事件函数 listmenu()，此时 content 被赋值使得下拉菜单被弹出，shownavindex 从 0 到 1 的变化使得一级菜单和向下箭头的样式也发生变化。

5.3.3 栏目切换场景

在文章管理或者商品管理小程序中，往往有分类或者栏目的切换，当点击某一栏（某一区域），显示区域的文章和商品被重置为被点击的栏目下的文章和商品。

例 5-3 本例设计商品栏目的切换效果，右侧显示区域根据左侧栏目的选择来显示对应的内容，运行效果如图 5.5 和图 5.6 所示。

图 5.5 “手机数码”栏目

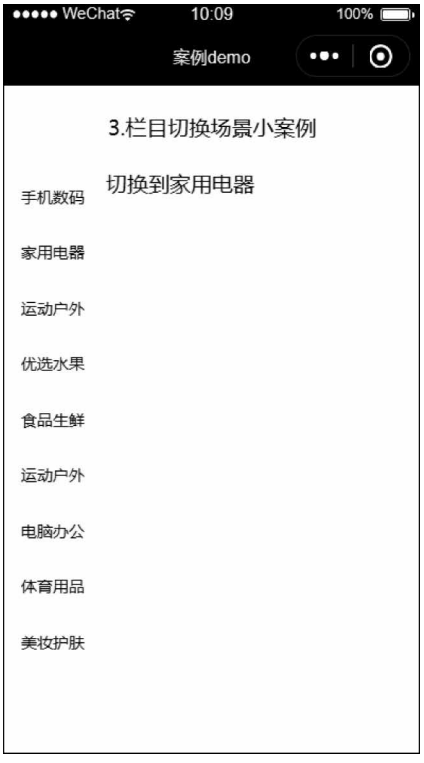

图 5.6 “家用电器”栏目

视频讲解

pages/switchTab/switchTab.wxml 文件代码如下：

```
<view class="title">3.栏目切换场景小案例</view>
<view class="content">
  <view class="left">
    <!--左侧部分-->
    <scroll-view scroll-y>
      <block wx:for="{{list1}}" wx:for-index="id">
        <view id="{{id}}" bindtap="switchTab">{{item}}</view>
      </block>
    </scroll-view>
  </view>
  <view class="right">
    <!--右侧部分-->
    <swiper current="{{currentTab}}" >
```

```
      <block wx:for="{{list2}}">
        <swiper-item>
          {{item}}
        </swiper-item>
      </block>
    </swiper>
  </view>
</view>
```

pages/switchTab/switchTab.js 文件代码如下：

```
Page({
  data: {
    currentTab: 0,                                    //初始化数据
    list1: ["手机数码", "家用电器", "运动户外", "优选水果", "食品生鲜", "运动户外", "电脑办公", "体育用品", "美妆护肤"],
    list2: ["切换到手机数码", "切换到家用电器", "切换到运动户外", "切换到优选水果", "切换到食品生鲜", "切换到运动户外", "切换到电脑办公", "切换到体育用品", "切换到美妆护肤"],
  },
  switchTab: function(e) {
    var that = this;
    var id = e.target.id;                             //获取 id
    if (this.data.currentTab == id) {
      return                                          //与 currentTab 值一致返回
    } else {
      that.setData({
        currentTab: id                                //设置 currentTab 值为 id
      });
    }
  }
})
```

pages/switchTab/switchTab.wxss 文件代码如下：

```
/*布局样式*/
.content {
  display: flex;
  flex-direction: row;
}
/*左边样式*/
.left {
  width: 25%;
  font-size: 30rpx;
}
scroll-view {
  height: 90%;
}
/*左边元素样式*/
.left view {
  text-align: center;
  height: 100rpx;
```

```
    line-height: 100rpx;
}
/* 右边样式 */
.right {
    width: 75%;
}
```

【代码讲解】 本例 switchTab.wxml 使用 flex 布局实现栏目和内容的左右分布，左侧是< scroll-view >组件，右侧是< swiper >组件，当用户点击左侧栏目的时候 switchTab.js 获取栏目的 id，并把 id 赋值给 current，此时右侧的< swiper >组件的第 id 项被显示，从而实现了左右的同步。本例巧妙地使用了< swiper >组件的 current 属性实现内容的切换。

5.3.4 系统设置场景

大部分的 App 和小程序都有系统设置功能，系统设置功能主要是用户输入或者选中数据，JavaScript 获取用户设置的数据来修改系统配置参数。系统设计场景是 JavaScript 在微信小程序中常见的应用场景。

视频讲解

例 5-4 本例以辩论赛小程序为背景，系统设置功能可以对立论、驳立论、质辩、自由辩论和总结陈词等环节的辩论时间和倒计时时间进行设置，系统设置页效果如图 5.7 所示，项目首页简单设计如图 5.8 所示。

图 5.7 设置页面

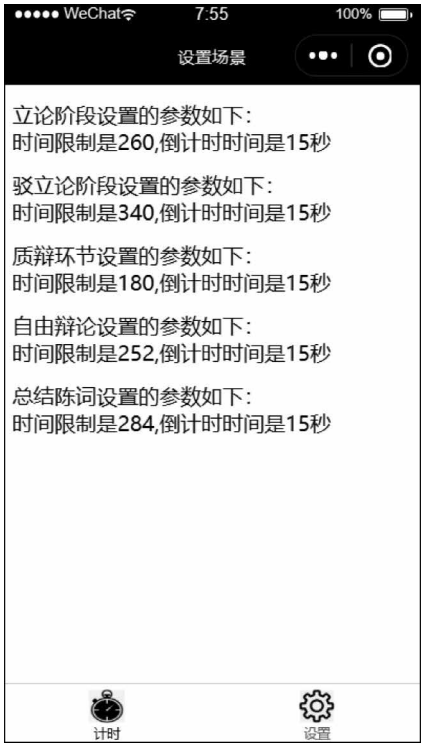

图 5.8 项目首页

app.js 文件代码如下：

```
App({
  data: { configs: [{ name: "立论阶段", time: 180, voice: 15 }, { name: "驳立论阶段", time: 180, voice: 15 }, { name: "质辩环节", time: 180, voice: 15 }, { name: "自由辩论", time: 180, voice: 15 }, { name: "总结陈词", time: 180, voice: 15 }]},
  onLaunch: function() {
    wx.setStorageSync('configs',this.data.configs);
  }
})
```

pages/setting/setting.wxml 文件代码如下：

```
<!-- 立论阶段 -->
<view>
  <view class="title">立论阶段</view>
  <view class="hr"></view>
  <view>
    <text>时间限制(秒)</text>
    <slider id="0" bindchange="sliderChange" show-value min="10" max="360" value="{{configs[0].time}}" />
  </view>
  <view>
    <text>声音提醒</text>
    <radio-group id="0" class="item" bindchange="radioChange">
      <label>
        <radio value="15" checked/>提前 15 秒</label>
      <label>
        <radio value="10" />提前 10 秒</label>
      <label>
        <radio value="5" />提前 5 秒</label>
    </radio-group>
  </view>
  <!-- 驳立论阶段 -->
  <view class="title">驳立论阶段</view>
  <view class="hr"></view>
  <view>
    <text>时间限制(秒)</text>
    <slider id="1" bindchange="sliderChange" show-value min="10" max="360"
    value="{{configs[1].time}}" />
  </view>
  <view>
    <text>声音提醒</text>
    <radio-group id="1" class="item" bindchange="radioChange">
      <label>
        <radio value="15" checked/>提前 15 秒</label>
      <label>
        <radio value="10" />提前 10 秒</label>
      <label>
        <radio value="5" />提前 5 秒</label>
    </radio-group>
```

```
  </view>
    <!-- 质辩环节 -->
  <view class="title">质辩环节</view>
  <view class="hr"></view>
  <view>
    <text>时间限制(秒)</text>
    <slider id="2" bindchange="sliderChange" show-value min="10" max="360"
    value="{{configs[2].time}}" />
  </view>
  <view>
    <text>声音提醒</text>
    <radio-group id="2" class="item" bindchange="radioChange">
      <label>
        <radio value="15" checked/>提前 15 秒</label>
      <label>
        <radio value="10" />提前 10 秒</label>
      <label>
        <radio value="5" />提前 5 秒</label>
    </radio-group>
  </view>
   <!-- 自由辩论 -->
  <view class="title">自由辩论</view>
  <view class="hr"></view>
  <view>
    <text>时间限制(秒)</text>
    <slider id="3" bindchange="sliderChange" show-value min="10" max="360"
    value="{{configs[3].time}}" />
  </view>
  <view>
    <text>声音提醒</text>
    <radio-group id="3" class="item" bindchange="radioChange">
      <label>
        <radio value="15" checked/>提前 15 秒</label>
      <label>
        <radio value="10" />提前 10 秒</label>
      <label>
        <radio value="5" />提前 5 秒</label>
    </radio-group>
  </view>
<!-- 总结陈词 -->
  <view class="title">总结陈词</view>
  <view class="hr"></view>
  <view>
    <text>时间限制(秒)</text>
    <slider id="4" bindchange="sliderChange" show-value min="10" max="360"
    value="{{configs[4].time}}" />
  </view>
  <view>
    <text>声音提醒</text>
    <radio-group id="4" class="item" bindchange="radioChange">
      <label>
```

```
        <radio value="15" checked/>提前 15 秒</label>
      <label>
        <radio value="10" />提前 10 秒</label>
      <label>
        <radio value="5" />提前 5 秒</label>
    </radio-group>
  </view>
</view>
```

pages/setting/setting.js 文件代码如下：

```
Page({
  data:{
    configs:[]
  },
  onLoad:function(options){
    var configs = wx.getStorageSync('configs');
    this.setData({configs:configs});
  },
  sliderChange:function(e){
    var id = e.target.id;
    this.data.configs[id].time = e.detail.value
    wx.setStorageSync('configs', this.data.configs);
    console.log(this.data.configs)
  },
  radioChange:function(e){
    var id = e.target.id;
    this.data.configs[id].voice = e.detail.value
    wx.setStorageSync('configs', this.data.configs);
    console.log(this.data.configs)
  }
})
```

pages/setting/setting.wxss 文件代码如下：

```
/*页面样式*/
page {
  background-color: skyblue; color: black;
}
/*标题样式*/
.title {
  font-size: 40rpx; height: 72rpx;
  line-height: 72rpx; padding-left: 20rpx;
}
/*分隔线样式*/
.hr {
  margin: 12px; width: 100%;
  height: 1px; background-color: grey;
}
/*文字样式*/
text {
  padding-left: 20rpx; font-size: 30rpx;
}
```

```
/* 选项样式 */
.item {
  margin: 12rpx;
}
/* 标签样式 */
.item label {
  margin: 10px;
}
```

pages/index/index.wxml 文件代码如下：

```
<view  wx:for="{{configs}}">
<view class="setitem"></view>
<view>{{item.name}}设置的参数如下：</view><view>时间限制是{{item.time}},倒计时时间是
{{item.voice}}秒</view>
</view>
```

pages/index/index.js 文件代码如下：

```
Page({
  data: {
    configs: []
  },
  onShow: function (options) {
    var configs = wx.getStorageSync('configs');
    this.setData({ configs: configs });
  },
})
```

pages/index/index.wxss 文件代码如下：

```
.setitem{
  height: 30rpx;
}
view {
  padding-left: 10rpx;
}
```

【代码讲解】 本例中辩论赛数据 configs 存放在 app.js 文件中，通过 wx.setStorageSync() 和 wx.getStorageSync()接口实现多个页面之间数据的共享和维护，setting 页面的< slider >和< radio >组件输入的数据被事件函数 sliderChange()和 radioChange()用来修改全局数据 configs。项目需注意修改全局数据 configs 时的下标问题。

5.4 实训项目——计算器小案例

视频讲解

本实训项目设计一个计算器，实现了加、减、乘、除运算。程序运行效果如图 5.6 所示。本项目没有涉及小程序复杂的接口，使用的是第 4 章样式与布局和第 5 章 JavaScript 基础的知识点。项目的设计思路是先编写 WXML 文件，然后编写 JS 接收 WXML 的数据的代

码，再编写加、减、乘、除运算的逻辑代码，最后编写 WXSS 文件。程序执行效果如图 5.9 所示。

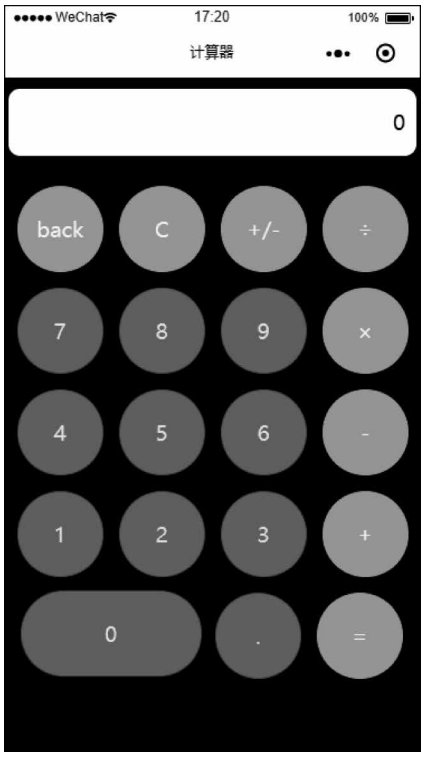

图 5.9　计算器示例图

pages/calculator/addList.wxml 文件代码如下：

```
<view class="demo-box">
  <!--输入框-->
  <view class="input-screen">{{screenData}}</view>
  <!--按钮-->
  <view class="btn-group">
    <view class="item green" bindtap="onclickButton" id="{{id1}}">back</view>
    <view class="item green" bindtap="onclickButton" id="{{id2}}">C</view>
    <view class="item green" bindtap="onclickButton" id="{{id3}}">+/-</view>
    <view class="item green" bindtap="onclickButton" id="{{id4}}">÷</view>
  </view>
  <view class="btn-group">
    <view class="item blue" bindtap="onclickButton" id="{{id5}}">7</view>
    <view class="item blue" bindtap="onclickButton" id="{{id6}}">8</view>
    <view class="item blue" bindtap="onclickButton" id="{{id7}}">9</view>
    <view class="item green" bindtap="onclickButton" id="{{id8}}">×</view>
  </view>
  <view class="btn-group">
    <view class="item blue" bindtap="onclickButton" id="{{id9}}">4</view>
    <view class="item blue" bindtap="onclickButton" id="{{id10}}">5</view>
    <view class="item blue" bindtap="onclickButton" id="{{id11}}">6</view>
```

```
    <view class="item green" bindtap="onclickButton" id="{{id12}}">-</view>
  </view>
  <view class="btn-group">
    <view class="item blue" bindtap="onclickButton" id="{{id13}}">1</view>
    <view class="item blue" bindtap="onclickButton" id="{{id14}}">2</view>
    <view class="item blue" bindtap="onclickButton" id="{{id15}}">3</view>
    <view class="item green" bindtap="onclickButton" id="{{id16}}">+</view>
  </view>
  <view class="btn-group">
    <view class="item1 blue" bindtap="onclickButton" id="{{id17}}">0</view>
    <view class="item blue" bindtap="onclickButton" id="{{id18}}">.</view>
    <view class="item green" bindtap="onclickButton" id="{{id19}}">=</view>
  </view>
</view>
```

pages/calculator/addList.js 文件代码如下：

```
Page({
  data: {
    id1: "back",
    id2: "clear",
    id3: "negative",
    id4: "÷",
    id5: "7",
    id6: "8",
    id7: "9",
    id8: "×",
    id9: "4",
    id10: "5",
    id11: "6",
    id12: "-",
    id13: "1",
    id14: "2",
    id15: "3",
    id16: "+",
    id17: "0",
    id18: ".",
    id19: "=",
    screenData: "0",                                    //屏幕数字初始化
    lastInput: false,                                   //控制输入
    array: [],                                          //定义一个空数组
  },
  onclickButton: function(e) {
    var id = e.target.id;
    if (id == this.data.id1) {                          //退格
      var data = this.data.screenData;
      if (data == 0) {
        return;
      }
      data = data.substring(0, data.length - 1);        //删除最后一位
      if (data == "" || data == "-") {                  //如果为空或者符号置零
```

```
      data = 0;
    }
    this.setData({
      screenData: data
    });
    this.data.array.pop();
    this.data.array.push(data);
  } else if (id == this.data.id2) {                    //清屏
    this.setData({
      screenData: "0"
    });
    this.data.array.length = 0;
  } else if (id == this.data.id3) {                    //正负号
    var data = this.data.screenData;
    if (data == 0) {
      return;
    }
    var firstWord = data.substring(0, 1);
    if (firstWord == "-") {
      data = data.substring(1, data.length);           //去掉负号
      this.data.array.shift();
    } else {
      data = "-" + data;
      this.data.array.unshift("-");                    //增加负号
    }
    this.setData({
      screenData: data
    });
  } else if (id == this.data.id19) {                   //=
    var data = this.data.screenData;
    if (data == 0) {
      return;
    }
    var lastWord = data.substring(data.length - 1, data.length);
    if (isNaN(lastWord)) {
      return;                                          //最后一位不是数字返回
    }
    var num = "";                                      //定义一个字符串
    var lastInput;
    var array = this.data.array;
    var optaration = [];
    for (var i in array) {
      if (isNaN(array[i]) == false || array[i] == this.data.id18 || array[i]
        == this.data.id3) {
        num += array[i];                               //对小数点或者正负号进行合并处理
      } else {                                         //加减乘除单独处理
        lastInput = array[i];
        optaration.push(num);
        optaration.push(array[i]);
        num = "";
      }
```

```
      }
      optaration.push(Number(num));
      var result = Number(optaration[0]) * 1.0;
      for (var i = 1; i < optaration.length; i++) {
        if (isNaN(optaration[i])) {
          if (optaration[1] == this.data.id4) {          //除运算
            result /= Number(optaration[i + 1]);
          } else if (optaration[1] == this.data.id8) {   //乘运算
            result *= Number(optaration[i + 1]);
          } else if (optaration[1] == this.data.id12) { //减运算
            result -= Number(optaration[i + 1]);
          } else if (optaration[1] == this.data.id16) { //加运算
            result += Number(optaration[i + 1]);
          }
        }
      }
      this.data.array.length = 0;
      this.data.array.push(result);
      this.setData({
        screenData: result + ""                          //将计算结果赋值用于在屏幕上显示
      });
    } else {
      if (id == this.data.id4 || id == this.data.id8 || id == this.data.id12 || id ==
this.data.id16) {
        if (this.data.lastInput == true || this.data.screenData == 0) {
          return;                                        //开始不允许输入加减乘除
        }
      }
      var vd = this.data.screenData;
      var data;
      if (vd == 0) {
        data = id;
      } else {
        data = vd + id;                                  //用于连续输入数字
      }
      this.setData({
        screenData: data
      });
      this.data.array.push(id);
      if (id == this.data.id4 || id == this.data.id8 || id == this.data.id12 || id ==
this.data.id16) {
        this.setData({
          lastInput: true
        });
      } else {
        this.setData({
          lastInput: false
        });
      }
    }
  }
```

```
})
```

pages/calculator/addList.wxss 文件代码如下：

```
/*外部布局*/
.demo-box{
  background-color: black; padding: 10rpx;
}
/*输入框样式 */
.input-screen{
  background-color: white; border-radius: 20rpx; text-align: right;
  width: 710rpx; height: 120rpx; line-height: 120rpx;
  padding-right: 20rpx; margin-bottom: 40rpx;
}
/*按钮布局方式*/
.btn-group{
  display: flex; flex-direction: row;
}
/*按钮样式*/
.item{
  width: 150rpx; height: 150rpx;
  margin: 15rpx; border-radius: 100rpx;
  text-align: center; line-height: 150rpx;
}
/*按钮 0 样式*/
.item1{
  width: 320rpx; height: 150rpx;
  margin: 10rpx; border-radius: 100rpx;
  text-align: center; line-height: 150rpx;
}
/*颜色样式*/
.green{
  color: #f7f7f7; background: #7ccd7c;
}
/*颜色样式*/
.blue{
  color: #f7f7f7; background: #0095cd;
}
```

【代码讲解】 本项目在 addList.wxml 文件和 addList.wxss 文件中完成页面的布局，为每个按钮绑定点击事件；在 JS 文件中首先设置允许连续输入操作数并且不允许连续输入操作符，然后分别为每个按钮设置交互功能，包括退格、清零和正负号，接着定义等号的交互，数字在连续输入时将中间结果保存在数组中，最后实现加减乘除运算。由于考虑到篇幅问题，设计功能完整的计算器代码较多，故没有涉及运算的优先级处理，有兴趣的读者可以自行修改本项目。