8.1 小程序基础库

我们在第三章提到过小程序的运行环境是分成渲染层和逻辑层的，我们在渲染层可以用各类组件组建界面的元素，在逻辑层可以用各类 API 来处理各种逻辑，这里提到的组件、API 其实都是小程序基础库进行包装提供的，基础库的职责还要处理数据绑定、组件系统、事件系统、通信系统等一系列框架逻辑，才能让整个小程序有序的运作起来。小程序的基础库是 JavaScript 编写的，它可以被注入到渲染层和逻辑层运行。为了叙述方便，下文我们把小程序基础库简称为基础库。

8.1.1 基础库载入时机

我们在开发网页时，经常会引用很多开源的 JS 库，在使用到这些库所提供的 API 前，我们需要先在业务代码前边引入这些库，如代码 8-1 所示，我们在使用 jQuery 库的$函数前，需要在业务代码前用 script 标签先引入 jQuery.js。

代码清单 8-1 在 HTML 引入其他 JS 库

```
<script src="jQuery.js"></script>
<script src="Vue.js"></script>
<script>
  $(function() {
        // console.log("ready")
  })
</script>
```

同样道理，为了让小程序业务代码能够调用 wx.navigateTo 等 API 以及组件，就需要在启动小程序后先载入基础库，接着再载入业务代码。由于小程序的渲染层和逻辑层是两个线程管理，渲染层 WebView 层注入的称为 WebView 基础库，逻辑层注入的称为 AppService 基础库，如果没有特殊声明，后文提到基础库的概念指的是 WebView 基础库和 AppService 基础库两个的组合。

显然，所有小程序在微信客户端打开的时候，都需要注入相同的基础库 ，所以，小程序的基础库不会被打包在某个小程序的代码包里边，它会被提前内置在微信客户端。这样做的好处有两点：

1. 降低业务小程序的代码包大小。
2. 可以单独修复基础库中的 Bug，无需修改到业务小程序的代码包。

8.1.2 基础库的版本号

小程序基础库版本号使用 semver 规范，格式为 Major.Minor.Patch，其中 Major、Minor、Patch 均为整数，1.9.901、2.44.322、10.32.44 都是符合 semver 风格的版本号。通常我们月度发布版本会把 Minor 提升一位，例如从 1.9.x 升级到 1.10.x，如果是修正版本，会把 Patch 提升一位，例如 1.10.0 升级到 1.10.1。Major 位则是重大特性发布时才会被提升一位。
在小程序中，可以通过 wx.getSystemInfo()或者 wx.getSystemInfoSync()方法获取小程序版本号，如代码 8-2 所示。

代码清单 8-2 用字符串直接比较版本号

```
var info = wx.getSystemInfoSync()
console.log("小程序基础库版本号为：" + info.SDKVersion)
```

我们还要再强调一点，不少开发者会使用错误的版本号比较方法，例如直接用字符串比较，parseInt 比较等，往后当基础库版本号提升上去后，会引发一些逻辑错误，如代码 8-3 所示。

代码清单 8-3 用字符串直接比较版本号

```
var info = wx.getSystemInfoSync() // info.SDKVersion == "1.11.0"
if (info.SDKVersion > "1.9.0") { // 此时条件为 false，无法进入 if 分支
  // 处理高版本小程序的逻辑
}
```

代码 8-4 给出了正确的比较版本号的方法，后续小程序基础库会内置版本比较的 API，建议查阅小程序官方文档使用该方法。

代码清单 8-4 正确比较版本号的方法

```
function compareVersion(v1, v2) {
  v1 = v1.split('.')
  v2 = v2.split('.')
  var len = Math.max(v1.length, v2.length)

  while (v1.length < len) {
    v1.push('0')
  }
  while (v2.length < len) {
    v2.push('0')
  }

  for (var i = 0; i < len; i++) {
    var num1 = parseInt(v1[i])
    var num2 = parseInt(v2[i])

    if (num1 > num2) {
      return 1
    } else if (num1 < num2) {
      return -1
    }
  }
  return 0
}

compareVersion('1.11.0', '1.9.9') // => 1 // 1 表示 1.11.0 比 1.9.9 要新
compareVersion('1.11.0', '1.11.0') // => 0 // 0 表示 1.11.0 和 1.11.0 是同一个版本
compareVersion('1.11.0', '1.99.0') // => -1 // -1 表示 1.11.0 比 1.99.0 要老
```