

第 3 章



感受 uni-app

本章将通过修改及调试默认模板项目为读者介绍 uni-app 项目的调试技术,并将项目重新编译打包为微信小程序能运行的包,以此来体验 uni-app 跨平台的便利性,之后将通过对比使用 uni-app 开发微信小程序及使用微信开发者工具的方式来为读者介绍 uni-app 对比原生开发微信小程序的优势,并通过这样的对比方式来向读者介绍 uni-app 中的技术原理。

3.1 Web 端运行调试 uni-app

在项目研发的过程中程序往往会不断地经历编码、编译、运行、调试这些阶段。在之前的章节中已经为读者介绍过编译和运行的概念,所谓编译是指将开发者编写的代码翻译成目标语言或者指令,经历多次翻译后最终变为计算机可以识别的二进制语言。所谓运行是指将程序运行起来,此时开发者可以一边修改代码一边立即看到修改后的结果,同时可以打印 log 日志。而调试,也称为 Debug,是指在运行的基础上,进一步可以打断点、单步跟踪、看堆栈信息,以此来排查及修复程序在运行过程中发生的错误。

uni-app 可以用 CLI 工具提供的 npm 命令对程序进行调试,但更重要的是 uni-app 的专用开发工具 HBuilder X,它提供的调试工具可以帮助开发者更好地开发 uni-app 项目。简要说来, HBuilder X 为 uni-app 提供了内置的 Web 浏览器、Web 端调试环境、App 的真机运行环境、App 调试环境、uniCloud 运行环境、uniCloud 调试环境,极大地提高了开发者的效率。本章将重点讲述 uni-app 项目在 HBuilder X 中在 Web 端、App 端及微信小程序平台中如何进行调试。

3.1.1 uni-app 在 Web 端调试运行

打开 uni-app 项目的页面,单击 HBuilder X 右上角的预览按钮,如果是第 1 次运行,则会提示需要下载内置浏览器,单击 Yes 按钮即可完成插件的安装,安装该插件时弹出的窗口如图 3-1 所示。

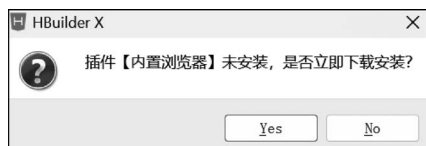


图 3-1 安装内置浏览器插件

编译运行成功后可以在内置浏览器里看到项目首页展示。该首页如图 3-2 所示。

如果程序在运行中修改并保存工程源码,则会触发自动编译,对应的浏览器所显示页面也会进行热加载。在 HBuilder X 控制台里,可以直接看到热加载时的日志,输出如图 3-3 所示。

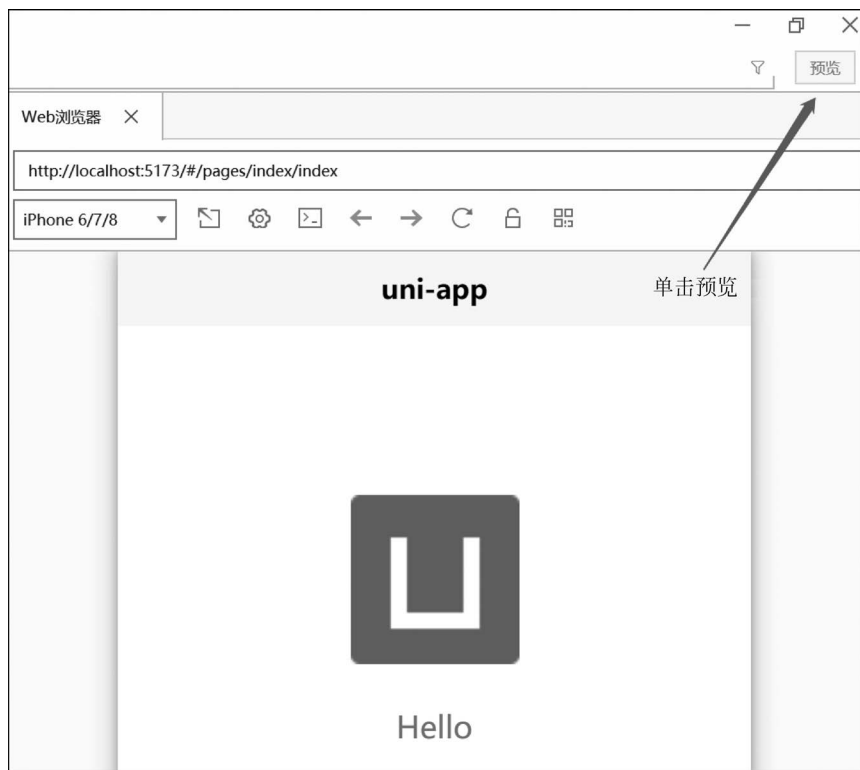


图 3-2 在内置浏览器中预览

```
App Launch at App.vue:4
App Show at App.vue:7
[vite] hot updated: /pages/index/index.vue
hmr update /pages/index/index.vue
```

图 3-3 HBuilder X 热更新

💡 **注意：** HBuilder X 3.3.3+ 编译器改为 vite，之前版本的编译器为 webpack。

通过单击控制台打印的日志，还可以直接跳转到对应的代码处，其操作如图 3-4 所示。

💡 **注意：** 浏览器控制台打印的日志无法转到代码，只有在 HBuilder X 控制台打印的日志才能转到代码，而且当运行到外部浏览器时也没有这个功能，只有通过 HBuilder X 内置浏览器才可以实现跳转功能。

3.1.2 uni-app 在 Web 端同步断点

HBuilder X 中有两种断点调试方案，一种是使用浏览器自带的调试控制台，另一种是使用 HBuilder X 的调试控制台。在 HBuilder X 内置浏览器中，首先单击“显示开发者工具”按钮，如图 3-5 所示，可以看到该内置浏览器的控制台。



4min



图 3-4 使用内置浏览器调试



图 3-5 打开内置浏览器控制台

之后在控制台中找到要调试的源码后右击,可以直接将断点同步到内置浏览器的调试工具,该操作如图 3-6 所示。

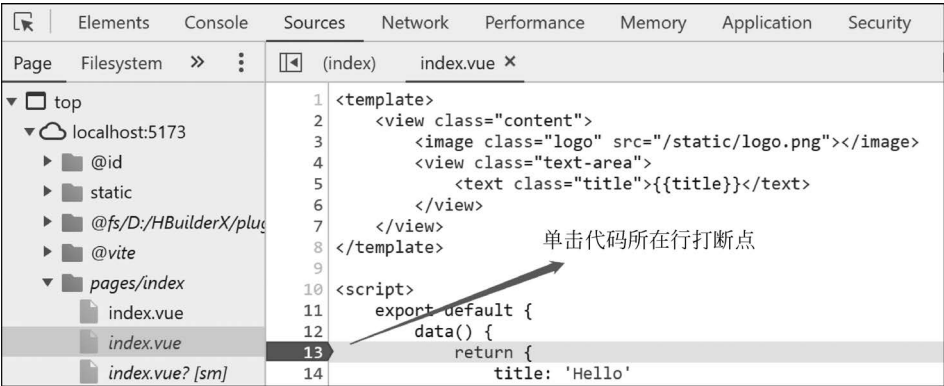


图 3-6 使用内置浏览器打断点

同样地,也可以使用外部浏览器对项目进行调试操作。首先将项目运行到外部浏览器,单击“运行”→“运行到浏览器”→选择对应浏览器,该操作如图 3-7 所示。



图 3-7 使用外部浏览器运行项目

运行成功后在如图 3-8 所示的位置编写 debugger 指令来告诉浏览器当运行到此处时需要断点进行断点同步。



图 3-8 通过 debugger 指令打断点

按下 F12 键打开浏览器开发模式,之后再按下 F5 键刷新页面,此时页面会重新渲染,当执行到 debugger 这条指令后浏览器就会进入断点状态,此时页面的显示如图 3-9 所示。

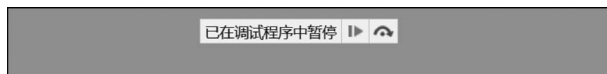


图 3-9 浏览器进入断点后的页面显示



3min

3.1.3 uni-app 在 Web 端中 Debug

uni-app 项目运行到 Web 后,支持在 HBuilder X 自带的调试面板中调试 JavaScript 代码。因为使用了 Chrome Debug 协议,在 Debug 过程中需要本机安装 Chrome 浏览器,调试支持的文件类型包括 Vue 文件、Nvue 文件、TypeScript 文件、JavaScript 文件,并且断点只能打在 JavaScript 或 TypeScript 代码中。

注意: 请勿在 template、style 等节点上添加断点,这会导致项目无法正常编译。

下面通过一个案例来介绍如何在 Web 端进行 Debug 操作。首先修改 index.vue 文件,代码如下:

```
//第2章/DebugIndex.vue
<template>
  <view class="content">
    <!-- 单击图片触发 hello 方法 -->
    <image class="logo" src="/static/logo.png" @click="hello()"></image>
    <view class="text-area">
      <text class="title">{{message}}</text>
```

```
</view>
</view>
</template>
<script>
  export default {
    data() {
      return {
        title: 'Hello',
        message: ''
      }
    },
    onLoad() {

    },
    methods: {
      //hello 方法会将 message 赋值为 Hello uni-app
      hello(){
        this.message = this.title + 'uni-app'
      }
    }
  }
</script>
```

修改之后的首页如图 3-10 所示。

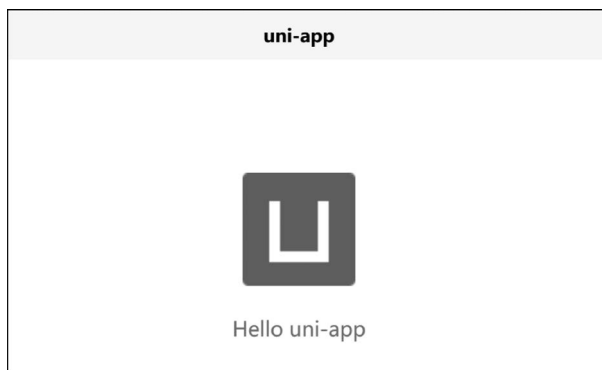


图 3-10 修改后的首页

通过单击 logo 图片触发 hello 方法,之后 message 的值通过字符串拼接最终变为 Hello uni-app,下面通过进入 Debug 模式来追踪 message 在程序中值的具体变化。首先单击 HBuilder X 工具控制台中的红色虫子按钮选择安装对应的 Debug 插件,该按钮如图 3-11 所示。



图 3-11 启用 Chrome 调试按钮

单击红色虫子按钮后选择开启 Chrome Debug 后会出现一个弹窗,显示是否安装 JavaScript 运行调试插件,单击“安装”按钮即可完成安装,该提示框如图 3-12 所示。



图 3-12 安装 Debug 插件

注意：如果使用外部浏览器对 uni-app 项目进行调试,则 HBuilder X 默认只支持 Chrome 浏览器。

与之前添加断点的方法相似,此时在需要同步断点的代码行双击即可添加或删除断点。在触发断点之后可以在“调试视图”中看到程序在运行过程中变量及其堆栈的变化信息,该调试视图页如图 3-13 所示。

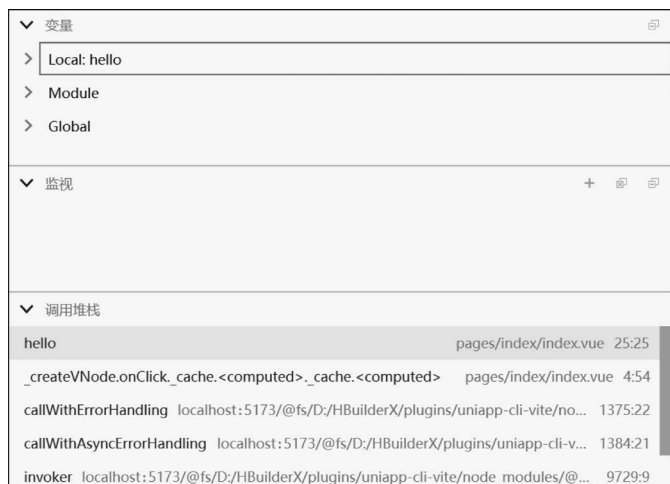


图 3-13 查看 Debug 视图

在 Debug 模式下将鼠标悬停在 message 变量上,可以看到 message 的具体值,如图 3-14 所示。



图 3-14 Debug 模式下查看变量值

而在调试视图的最上方可以进行相应的断点操作,如图 3-15 所示。



图 3-15 调试操作

这 4 个按钮分别对应以下操作。

1. 继续操作

对应快捷键 F8,继续操作代表进入下一个断点,这个操作应该是在调试过程中最常用的。

2. 下一步操作

对应快捷键 F10,下一步操作代表执行下一条语句,如果不想看当前方法内部执行的具体过程,则可以使用该操作,它会跳过当前的方法并进入下一种方法。

3. 进入操作

对应快捷键 F11,进入操作则代表查看该方法的内部执行情况,如果想看当前方法的具体内部实现,则可以使用此操作。

4. 返回操作

对应快捷键 Shift+F11,代表继续执行当前的方法直到最后一行,当在调试过程中进入一个嵌套调用但又不想看这个函数具体的内部实现时,该快捷键可以让断点直接进入下一个函数。

除了可以通过鼠标悬停的方式查看变量的具体值,还可以通过对变量添加监视来检测数据,查看变量,这个的操作如图 3-16 所示。



图 3-16 对变量添加监视

在添加完成之后就可以在监视栏中看到这个变量的具体值了,其显示值如图 3-17 所示。

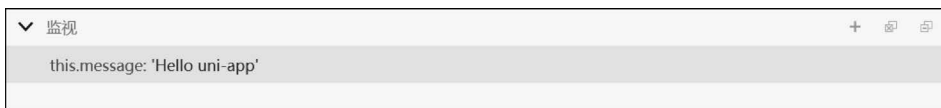


图 3-17 在监视栏中观察变量

以上讲述的内容基本覆盖了在开发 Web 端时所需要的调试技术。当然 HBuilder X 支持的调试功能不仅于此。在 3.2 节中会介绍如何在 HBuilder X 中进行真机运行调试。

3.2 真机运行调试 uni-app

所谓的“真机运行”是指使用真实的手机或手机模拟器来连接,以此进行测试,在 Android 平台 HBuilder X 支持安卓调试桥接(Android Debug Bridge,ADB)协议,在 HBuilder X 运行的计算机上,可以使用 USB 线连接 Android 设备,也可以使用安装在计算机上的 Android 模拟器(包括谷歌官方模拟器或者第三方模拟器如“雷电”“夜神”等),而真机运行的目的,是为了实现代码修改的热刷新,避免打包后才能看到效果。开发者通过调试在 HBuilder X 中的代码,就能在手机上实时看到修改效果,并且可以在 HBuilder X 控制台看到相关的日志信息。



3.2.1 运行到 Android 基座

选中项目之后通过 HBuilder X 顶部运行菜单来打开运行入口,选择“运行”→“运行到手机或模拟器”→“运行到 Android App 基座”,该操作如图 3-18 所示。



图 3-18 通过运行选项运行到 Android App 基座

也可以通过单击菜单栏上的运行按钮来选择“运行到 Android App 基座”,如图 3-19 所示。

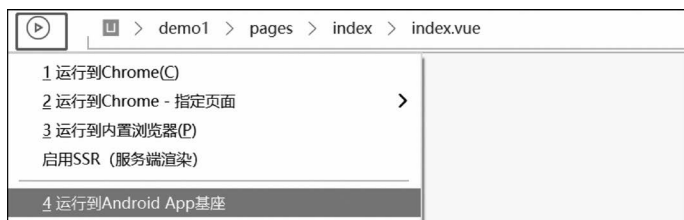


图 3-19 通过运行按钮运行到 Android App 基座

相比于通过 HBuilder X 顶部运行菜单,工具栏运行按钮下可选的内容较少,只保留了最常用的运行入口。

当单击运行到 iOS 或 Android 设备时会弹出设备选择的界面,需选择将要连接的手机设备或模拟器。可以多设备运行,每个运行设备会在 HBuilder X 底部控制台新开一个独立窗口,互不干扰,但一个设备同时只能运行一个项目,不同的项目运行到相同手机只有最后一个项目生效,该设备选择界面如图 3-20 所示。



图 3-20 选择 Android 设备

因手机差异较大,HBuilder X 并没有提供 App 的模拟器。下面介绍如何使用 Android 模拟器和 Android 手机进行调试。

3.2.2 使用 Android 模拟器运行调试

谷歌官方有自己的模拟器,可以在 Android Studio 中安装不同版本的模拟器,当然市场上也有很多成熟的第三方 Android 模拟器,这里就不推荐了,可自行搜索安装。在安装好后保持模拟器打开状态,单击刷新后会自动获取该设备,如图 3-21 所示。



图 3-21 获取 Android 模拟器信息

检测到模拟器之后选择“使用标准基座运行”,单击“运行”按钮。注意这里选择标准运行基座,是 DCloud 为了方便开发者低门槛调试而提供的,此基座 App 使用的是 DCloud 的包名、证书和第三方 SDK 配置。在原生层不变的情况下,JavaScript 等动态代码可以在基座上动态加载,以实现热重载运行。如果选择自定义基座,则表示需要开发者自定义原生层,需要执行一遍 iOS 或 Android 的打包流程,由 XCode 或 Android Studio 编译打包生成 IPA 或 APK 安装包,但打包后无法方便调试,不能热重载和显示控制台日志。

单击“运行”按钮,项目在经过编译后会自动打开模拟器的 ADB 文件完成连接,如果是第 1 次连接,则会显示下载调试基座及需要应用授权等信息,确认之后重新运行即可。模拟器连接过程中的日志输出如图 3-22 所示。



图 3-22 HBuilder X 连接到 Android 模拟器

连接完成之后,在模拟器中可以看到如图 3-23 所示的项目首页。

3.2.3 使用 Android 手机运行调试

首先确认 Android 手机设置中 USB 调试模式已开启。通常该模式可以在手机的设置中找到,部分手机在插上数据线后在系统通知栏里也可以设置,注意不能设置为 U 盘模式,如果是充电模式,则必须同时设置充电时允许 USB 调试。不同的机型对应的打开开发者模式的方法也有所不同,这里无法一一列举。连接之后模拟器会显示相关授权选项,单击允许后可以看到如图 3-24 所示的设备信息。

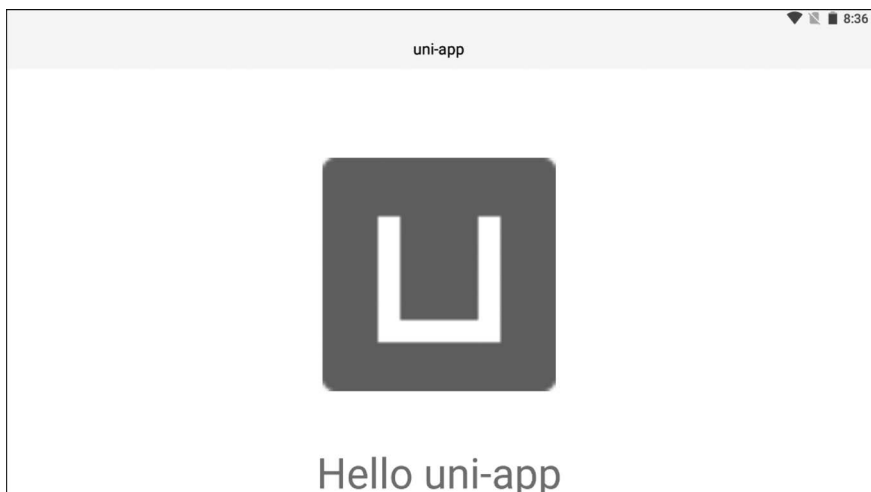


图 3-23 Android 模拟器运行项目

之后单击运行,和运行在模拟器中相同,在 HBuilder X 的控制台中可以看到这个连接过程的日志输出,日志打印的信息如图 3-25 所示。



图 3-24 获取 Android 设备信息

```
ready in 1403ms.  
正在建立手机连接...  
检测到HBuilder调试基座有更新,正在删除旧版本...  
删除旧版HBuilder调试基座成功  
正在安装手机端HBuilder调试基座...  
安装HBuilder调试基座完成  
正在同步手机端程序文件...  
同步手机端程序文件完成  
正在启动HBuilder调试基座...  
应用【demo1】已启动
```

图 3-25 连接到 Android 手机日志输出

同样地,在手机设备上也可以看到该项目的首页,如图 3-26 所示。

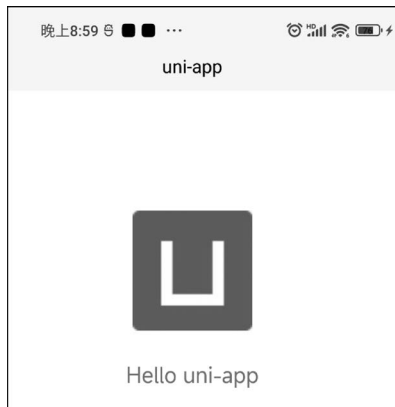


图 3-26 项目运行到 Android 手机

3.2.4 uni-app 在 Android 系统中 Debug

和在 Web 端进行 Debug 操作类似,首先需要单击控制台右侧的红色虫子按钮,此时会提示需要下载 uni-appApp 端调试插件,下载完毕后单击重新运行就可以在 HBuilder X 中调试手机端应用了,其具体操作如图 3-27 所示。



图 3-27 在 Android 系统中进行调试连接

如果使用 Android 手机连接到调试服务,则需要注意以下几点:

- (1) Android 手机和 PC 需要处于同一局域网。
- (2) Android 手机使用移动网络会导致无法连接调试服务。
- (3) Android 手机使用 VPN 等代理设置会导致无法连接调试服务。
- (4) PC 设置了防火墙。

在安装完插件并进入调试模式后,HBuilder X 中会弹出一个调试工具窗口。和浏览器自带的开发工具类似,该窗口允许开发者添加断点、查看日志、使用快捷键进行调试操作等,该工具窗口如图 3-28 所示。



图 3-28 HBuilder X 内置 Debug 插件窗口

以上的内容基本覆盖了在开发 uni-app 项目的 App 端时所常用的调试技术。这里需要特别注意的就是在连接 Android 手机进行调试时可能会遇到连接不上的情况。具体需要注意以下几点:

- (1) 确保计算机已安装相应的手机驱动。
- (2) 确保数据线或 USB 口正常,可替换不同的线或口来验证。
- (3) 确认 Android 手机设置中 USB 调试模式已开启,注意不能设置为 U 盘模式,如果被设置为充电模式,则必须同时设置充电时允许 USB 调试。
- (4) 如手机屏幕弹出需信任本计算机的询问,则需同意该授权。
- (5) Android 5.0 及以上系统,不要使用访客模式,在这种模式下无法成功运行。
- (6) 部分手机有 USB 安装应用的权限设置,需要在手机上允许通过 USB 安装应用。

由于不同厂商的手机/模拟器差异较大,在连接过程中可能会遇到不同的问题,在这里无法一一列举。如果在操作过程中遇到了难以解决的问题,则除了用搜索引擎寻求解答和参阅

官方资料外,还可以通过 uni-app 官方社区(<https://ask.dcloud.net.cn/explore/>)寻求解答。

3.3 uni-app 一键跨平台发布到微信小程序

本节将为读者介绍 uni-app 如何通过 HBuilder X 工具跨平台发布到微信小程序的具体操作。

3.3.1 配置 AppID 生成微信小程序项目

其实在小程序平台运行调试 uni-app 项目的过程中,uni-app 的项目代码已经被转换为微信开发者工具所能识别的代码。事实上跨平台发布与运行调试的过程类似。首先在 HBuilder X 的菜单栏中选择“发行”→“小程序-微信”,具体操作如图 3-29 所示。

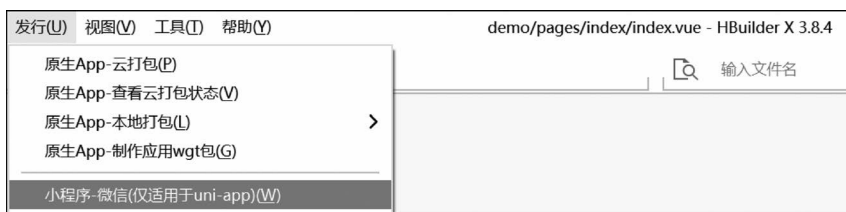


图 3-29 发布到微信小程序选项

如果发布时提示缺少 uni-app 标识 ID,则需要依据提示到官网进行用户注册并进行登录,登录到 HBuilder X 之后可以在 manifest.json 文件中获取 uni-app 的 AppID,该页面如图 3-30 所示。



图 3-30 获取 uni-app 的 AppID

在单击微信小程序之后的弹窗中,需要填写小程序对应的 AppID,不过当前测试项目可以随意填。在后面小程序发布的阶段会为读者介绍申请小程序 AppID 的具体流程。微信小程序发行窗口如图 3-31 所示。

单击“发行”按钮,项目会被重新编译,编译完成后在项目的 unpackage 目录下会生成一个名为 mp-weixin 的文件夹,该文件夹的具体位置如图 3-32 所示。



图 3-31 发布到微信小程序操作窗口



图 3-32 发布生成微信小程序项目

3.3.2 项目导入并运行到微信开发者工具

再来看导入并运行到微信开发者工具的具体操作：首先在微信开发者工具中单击“导入”按钮，选择名为 mp-weixin 的文件夹，之后单击测试号，如果以游客模式登录该 AppID，则值 touristappid 会被自动填入 AppID 中，该导入页面如图 3-33 所示。



1min

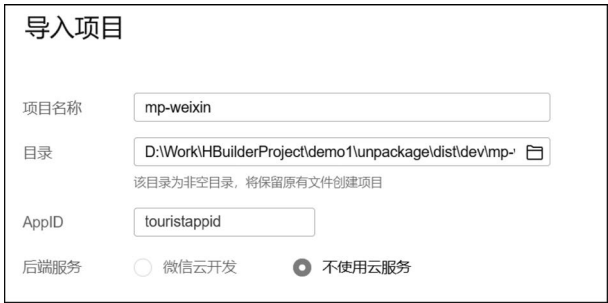


图 3-33 导入微信小程序项目

单击确定之后会出现是否信任项目并运行的界面,选择“信任并运行”,项目就会被编译并运行到微信开发者工具,该界面如图 3-34 所示。

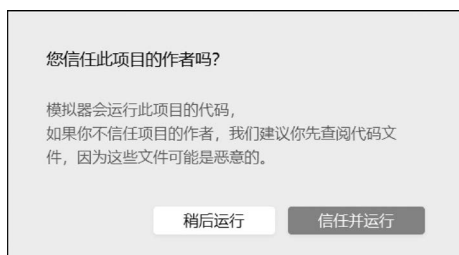


图 3-34 微信小程序运行

单击“信任并运行”按钮后,项目会被编译并运行,该项目的运行页面如图 3-35 所示。

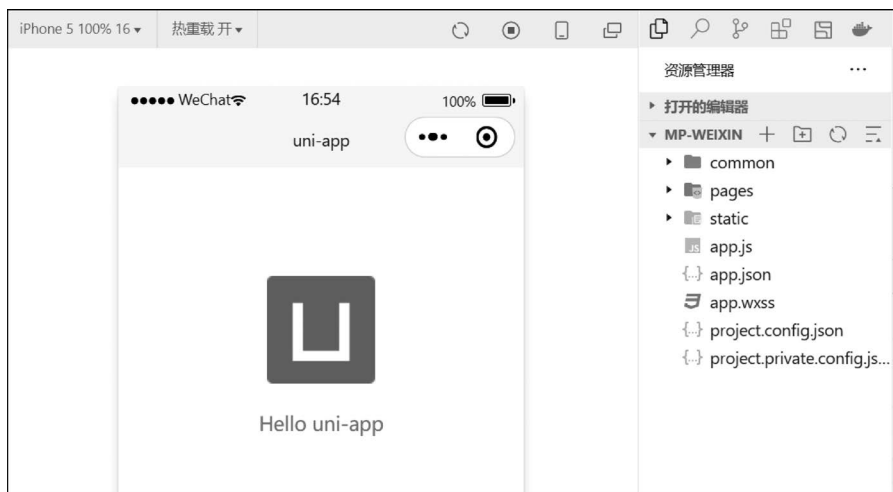


图 3-35 项目运行到微信开发者工具

3.4 微信开发者工具调试

在本节中将介绍如何在 HBuilder X 中将 uni-app 项目导入微信开发者工具并进行调试的相关操作。

3.4.1 通过 HBuilder X 运行到微信开发者工具

在微信小程序中的调试操作和 Web 端、App 端调试操作类似,选择“运行”→“运行到小程序模拟器”→“微信开发者工具”,如图 3-36 所示。

选择“微信开发者工具”后会出现微信开发者工具路径的配置界面,如图 3-37 所示。

通过该弹窗底部的 URL 网址可以进入微信开发者工具的下载页,下载对应系统的最新稳定版即可。配置完成后,单击“确定”按钮后项目会重新开始编译,同时在日志输出的控制台区域会出现名为小程序-微信的标签页,该标签页及具体的日志输出信息如图 3-38 所示。





图 3-36 运行到微信开发者工具



图 3-37 微信开发者工具路径的配置界面

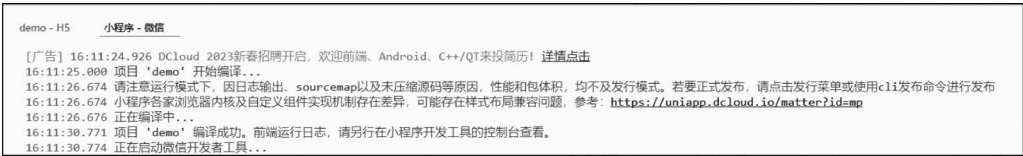


图 3-38 小程序-微信标签页日志输出

在编译的过程中会提示要打开微信开发者工具的服务器端口,可以通过在控制台输入命令或者打开微信开发者工具手动打开服务器端口。手动打开的方式: 打开微信开发者工具并单击右上方齿轮按钮,打开微信设置,该设置页面如图 3-39 所示。

之后会出现设置页面。单击“安全”按钮,并在出现的配置页面中打开服务器端口,该操作如图 3-40 所示。

打开服务器端口再次运行,此时会直接打开微信开发者工具并运行该项目,并且在HBuilder X中修改并保存的文件会自动地被刷新到微信模拟器。当前项目运行到微信开发者工具的首页,如图 3-41 所示。



图 3-39 打开微信开发者工具服务器端口



图 3-40 打开微信开发者工具服务器端口

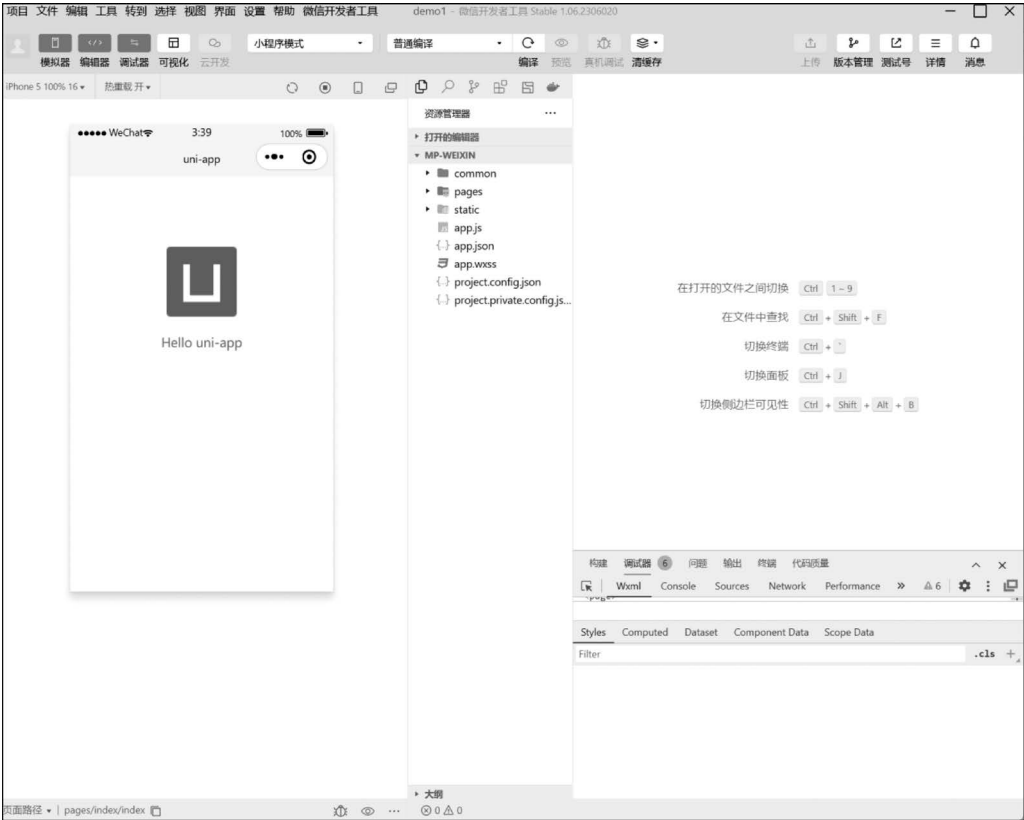


图 3-41 微信开发者工具首页

3.4.2 微信开发者工具调试

通过微信开发者工具调试页面的样式和一般的 Web 项目的调试操作类似,调试器 Wxml 栏下通过调试的箭头选中元素即可查看相应的节点和样式,该操作如图 3-42 所示。

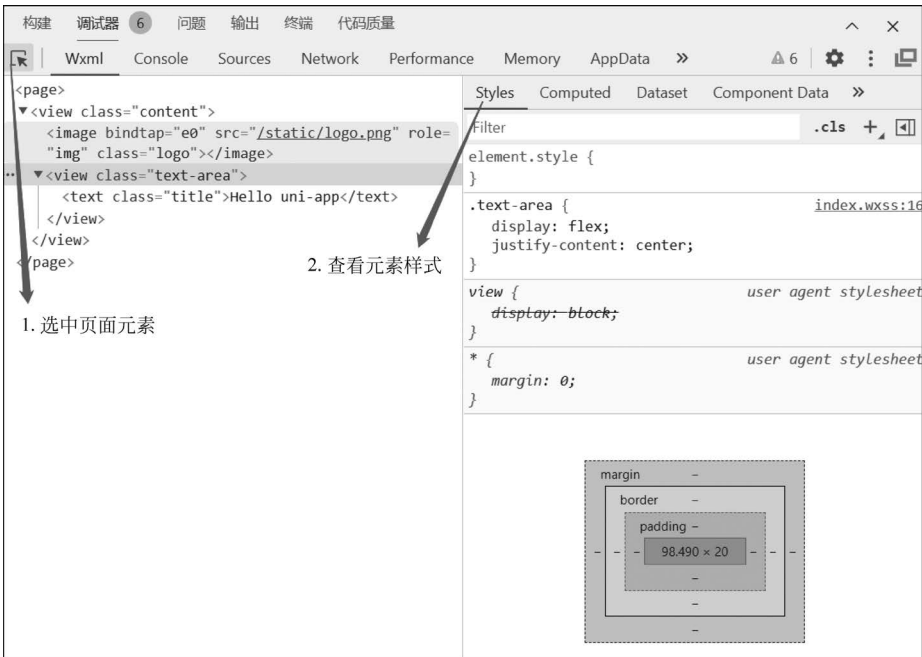


图 3-42 使用微信开发者工具查看页面元素

在调试 JavaScript 代码时需要切换到 Sources 栏,在 appContext 中找到对应的页面路径并选中想要调试的 JavaScript 代码进行调试(如果 JavaScript 代码被压缩过,则可单击右下角的{}格式化代码),该操作如图 3-43 所示。

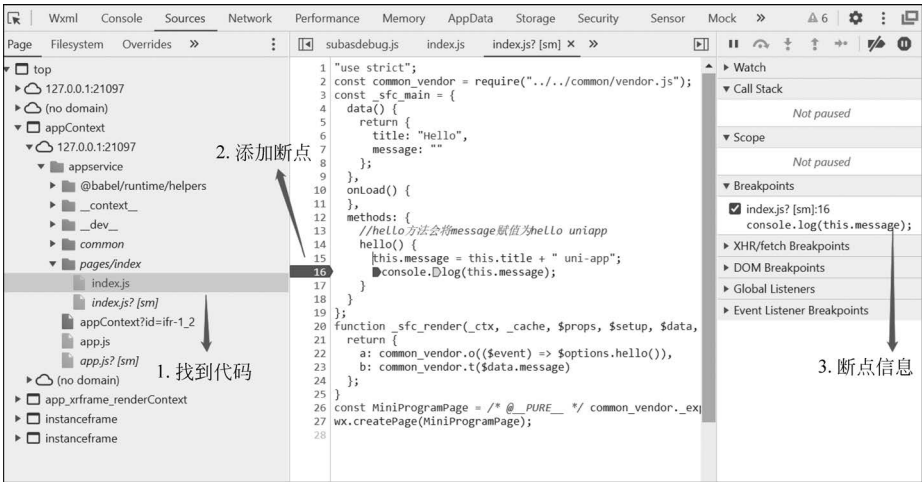


图 3-43 使用微信开发者工具调试

上述即为 uni-app 项目跨平台发布到微信小程序的具体流程,以及在微信开发者工具中如何进行调试操作。在实际操作过程中,因为考虑到不同平台下的兼容问题,所以在每个模块功能编写完成之后最好在不同平台提供的开发者工具下进行调试自检,以保证编写的代码在各个平台下所呈现的效果是一致的。在 3.5 节将会对比 uni-app 开发微信小程序与微信小程序原生开发 uni-app 的优势,以此来进一步介绍 HBuilder X 是如何“帮助”开发者高效率、高质量完成 uni-app 开发的。

3.5 uni-app 开发微信小程序与微信小程序原生开发对比

本节将从功能实现、性能体验、社区生态、开发体验、未来扩展性这 5 个方面来对比 uni-app 开发小程序和微信原生开发,并通过这种对比方式来进一步介绍在开发过程中 uni-app 是如何提高开发者的效率的。微信原生开发小程序有不少糟心的地方:

(1) 很多能提升性能和效率的工具(例如 node、webpack、预编译器 etc)对原生 wxml 开发的支持不太好,这势必会影响开发效率及工程的构建流程,所以无论是公司还是个人开发者都愿意使用框架进行开发。

(2) 微信小程序开发使用了私有语法,相较于主流 Vue.js 框架这套私有语法的通用性较差。

(3) 基于 Vue.js 开发规范的 uni-app 有非常丰富且专业的开发工具,而微信开发者工具对于这些高效的工具要略逊一筹。作为开发者,有时除了微信小程序,往往还要兼顾 Web 开发、其他小程序平台开发甚至是 App 开发,如果来回切换开发工具和语法变更,则势必会给开发人员带来极大的负担。选择跨平台框架自然可以解决这些问题,但是使用框架开发之后开发者又会有以下顾虑:

- ① 使用了 uni-app 跨平台框架之后,无法实现微信小程序中的某些功能。
- ② 使用了框架之后其性能不及原生开发。
- ③ 社区生态不完善,出现问题无法及时解决。
- ④ 开发体验上手难。
- ⑤ 如果项目需要跨平台发布,则可能存在扩展性问题。

下面就以这 5 个问题作为出发点逐个进行分析比对。

3.5.1 功能实现对比

先来设想这样一个问题,如果小程序迭代升级,新增了一批 API,如果 uni-app 框架未及时更新,则会对功能实现有影响吗?其实这是误解,uni-app 不限制底层 API 的调用。在小程序端中 uni-app 支持直接编写微信原生代码。对比于传统 Web 开发框架的使用,如果开发者无法使用浏览器提供的 JavaScript API,则这样的框架肯定是不成熟的且难以商用,而 uni-app 作为一款成熟的框架,可以调用微信小程序所提供的所有原生代码。举个例子,目前 uni-app 虽然尚未封装跨平台的广告组件,但在小程序端依然可以使用微信<ad>组件展现广告,代码如下:

```
//第3章/ad.vue
<view>
  <view class="title">微信官方 banner 广告</view>
    <view style="min-height: 50px;">
      <!-- uni-app 未封装,直接使用微信原生的 ad 组件 -->
      <ad unit-id="adunit-xxxxxx"></ad>
    </view>
  </view>
```

当然不仅是小程序广告组件,还包括微信小程序自定义组件、WXS、云开发等复杂功能,在 uni-app 里一样都是支持的,所以使用 uni-app 框架开发小程序在功能上和原生开发没有区别,不会有任何限制。

3.5.2 性能体验对比

为了屏蔽非必要的细节让开发者能够投入更多精力在解决业务问题上,相较于原生开发 uni-app 的框架内部做了层层封装,但作为开发者可能会疑问封装的框架是否会增加运行负载,从而导致性能下降,但实际上使用框架开发不仅能有效地提高开发效率,而且性能往往不会比原生开发差,类似于使用 Vue.js 开发 Web App,对比 JavaScript 进行原生开发,不但不会造成性能下降,反而由于虚拟 DOM 和增量更新技术的运用,在大多数场景下,比开发者手动写代码操作 DOM 的性能要好。同样,uni-app 作为一款成熟的框架不会导致性能下降,因为在很多环节做了自动优化,所以在绝大多数场景下其性能比微信原生开发更好,而在小程序中需要频繁地写 setData 代码来更新数据,而且要做增量数据更新。如果不做增量,则代码性能不好,但是每处逻辑都判断增量数据更新,那代码又会显得比较冗余,而使用 uni-app,底层自动进行增量数据更新,简单而高性能。下面以 setData 数据更新方法来展开性能优化的详细说明。

这里引用微信官方的描述,简单介绍 setData 背后的工作原理:小程序的视图层目前使用 WebView 作为渲染载体,而逻辑层是由独立的 JavaScriptCore 作为运行环境。在架构上,WebView 和 JavaScriptCore 都是独立的模块,并不具备数据直接共享的通道。当前,视图层和逻辑层的数据传输,实际上通过两边提供的 evaluateJavascript 所实现。

简单地说就是微信将 evaluateJavascript 调用封装成了 setData(Javascript)方法,从而实现视图层和逻辑层的数据传输,数据流如图 3-44 所示。

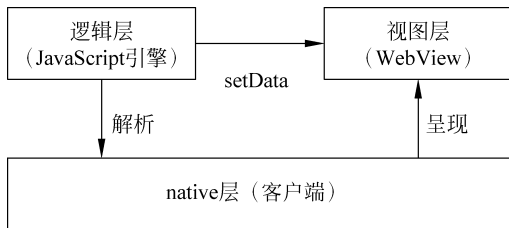


图 3-44 微信小程序 setData

setData 的执行会受到很多因素的影响,当 setData 每次传递数据量过大或频繁被调用时都可能引发性能问题,而针对这些问题 uni-app 进行了优化处理。

1. 减少 setData 传递数据量

假设当前页面有一个列表(初始值为 a、b、c),现在要向列表后追加 3 个新的列表项(d、e、f),在小程序中的代码如下:

```
//第3章/wxSetData.js
page({
  data:{
    list:['a','b','c']
  },
  add:function(){
    let addData = ['d','e','f'];
    this.data.list.push(...newData);
    this.setData({
      list:this.data.list
    })
  }
})
```

在 add 方法被执行时,会将 list 中的 a、b、c、d、e、f 这 6 个列表项通过 setData 全部传输过去,而在 uni-app 中这段代码如下:

```
//第3章/uniappSetData.js
export default{
  data(){
    return {
      list:['a','b','c']
    }
  },
  methods:{
    add:function(){
      let addData = ['d','e','f'];
      this.list.push(...newData)
    }
  }
}
```

可以看到在 uni-app 中执行 add 方法时,仅会将 list 中的 d、e、f 这 3 个新增列表项传输过去,大大简化了 setData 的传输量。uni-app 借鉴了 westore 中 JSON Diff 库的方法,在调用 setData 之前,会先比对历史数据,首先计算出有变化的增量数据,然后调用 setData,仅传输变化的数据,这样的增量传递可以极大地减少 setData 传输的数据。

2. 减少 setData 的调用频次

假设现在需要改变多个变量值,微信原生开发代码如下:

```
change:function(){
  this.setData({a:1});
  this.setData({b:2});
  this.setData({c:3});
}
```

4 次调用 setData 改变变量就会引发多次逻辑层、视图层之间的数据通信,而在 uni-app 中的代码如下:

```
change:function(){  
  this.a = 1;  
  this.b = 2;  
  this.c = 3;  
}
```

从原生开发的多次操作被改为一条数据,最终仅调用一次 setData 即可完成赋值操作,从而大幅度地降低了 setData 的调用次数。

3.5.3 社区生态对比

小程序由于是自造的生态,其脱离了 Web 应用的相关标准,所以有很多 Web 生态中的轮子无法使用,而相对于小程序,uni-app 的周边生态非常丰富,其插件市场有近上千插件,而且 uni-app 兼容小程序的生态,各种自定义组件均可直接引入并使用。在此基础上,uni-app 的插件市场还有更多的 Vue.js 组件,同时可跨多端使用,并且性能优秀。这使 uni-app 的生态成为最丰富的小程序开发生态。例如富文本解析、图表等组件,uni-app 的插件性能均超过了 wxparse、wx-echart 等微信小程序组件。在社区方面 uni-app 官方有 70 个开发者 QQ/微信交流群(大多为 2 千人群,共近 10 万开发者),而且还有为数众多的第三方群。问答社区每天有数百篇帖子。其活跃度与微信小程序官方论坛相同,远超过其他小程序官方论坛。不仅如此,uni-app 的第三方培训也非常活跃,腾讯课堂官方都为 uni-app 制作了课程,各种培训网站到处可见免费或收费的 uni-app 培训视频教程,所以综上所述,在社区生态方面,uni-app 显然要略胜一筹。

3.5.4 开发体验对比

微信原生的开发语法,它既像 React Native,又像 Vue.js,对于开发者来讲,需要学习一套新的语法,这无疑会大幅提高学习成本,而 uni-app 则对开发者更为友好,它的语法简单来讲是 Vue.js 的语法+小程序的 API。它遵循 Vue.js 语法规则,组件和 API 遵循微信小程序命名,这些都属于主流的通用技术,学习它们是前端必备技能,uni-app 没有太多额外的学习成本。也就是说有一定 Vue.js 基础和微信小程序开发经验的开发者可快速上手 uni-app,并且 Vue.js 也不需要全部掌握,只需了解 Vue.js 基础语法、数据绑定、列表渲染、组件等,其他如路由、Loader、CLI、Node.js、Webpack 并不需要学,因为 HBuilder X 工具搭配 uni-app 可以免终端开发,可视化创建项目、可视化安装组件和扩展编译器,所以 uni-app 的学习门槛非常低,而且其语法几乎和 Vue.js 一致,同时 HBuilder X 为开发者优化了部分细节,所以可以说使用 uni-app 开发 Web 难度甚至比用 Vue.js 还低。

而在开发体验层面,微信原生开发相比 uni-app 有较大差距,主要体现在:

(1) uni-app 拥有更为强大的组件化开发能力,得益于 Vue.js 对组件开发的支持,相比小程序自定义组件开发的体验要好很多。

(2) uni-app 支持 Vuex 进行应用状态管理。

(3) uni-app 支持使用 SASS 等 CSS 预处理器。

(4) uni-app 支持完整的 ES Next 语法。

(5) uni-app 中可以使用自定义构建策略(可选择不同的模板进行开发)。

而开发工具对比,两者的差距更大,微信开发者工具被吐槽无数,而 uni-app 的出品公司 DCloud 旗下的 HBuilder/HBuilder X 系列已是四大主流前端开发工具之一,而且其为 uni-app 应用开发做了很多优化,所以使用 HBuilder X 开发 uni-app 的效率、易用性非微信原生开发可及,所以综上所述,uni-app 的工程化能力要远大于微信原生开发。

3.5.5 扩展性对比

uni-app 的跨端功能极大地扩展了程序员的边界,基于 uni-app 开发的小程序无须修改即可同时发布为多家小程序,甚至是 App、H5 平台。只需一套源码就可以实现多端运行,而基于 uni-app 发展出的 uni-Cloud(云服务)还有 uni-AI(人工智能服务)等生态也在不断地壮大,相信未来的 uni-app 其拓展的边界将会越来越广阔。

3.6 本章小结

本章着重介绍了 uni-app 在 Web 端调试、真机调试、微信小程序端调试的相关操作,并通过 HBuilder X 工具将 uni-app 应用跨平台发布到微信小程序。最后又从 5 方面对比了 uni-app 开发小程序及原生开发小程序的优势。通过本章的内容,相信各位读者已经掌握了 uni-app 相关的调试技术及通过 HBuilder X 工具进行跨平台发布。自此已经介绍完开发 uni-app 所需的基本概念及基础知识。在接下来的章节中将会进行案例项目的开发,借此来进一步学习如何使用 uni-app 构建出完整应用。