基于 Weex 跨平台开发的研究与实践

—以 APP<微阅读>的开发为例

软件学院　　数字媒体技术
123022013062　刘婷　　指导教师　　张金颖

【摘　要】基于 Weex 跨平台开发框架，开发者仅使用 HTML、CSS、JavaScript 即可开发出 Android、iOS 原生应用，开启了跨平台开发新的里程碑。微阅读是一款基于 Weex 跨平台开发框架，同时使用 WebStorm 与 Android Studio 等工具开发，可在 Web 及 Android 平台上运行的应用。论文从软件开发角度，分析对比了使用 Weex 开发与原生开发的异同，详细描述了基于 Weex 的微阅读系统的需求分析，项目结构设计，项目架构设计，项目具体实现等，最后对系统进行了测试，结果表明基于 Weex，可成功实现代码一次编写，多端复用。

【关键词】跨平台开发；Weex 框架；Android

目 录

1 绪论

1.1 研究背景

目前，市场上大多数 app 都采用原生开发，这样开发一个项目，但就 Android，iOS，Web 来说，就需要三种不同的团队[1]，而且，对于 Android 和 iOS 来说，还存在开发周期长、迭代周期长等问题。而采用跨平台开发[2]，仅一套代码即可实现在 Web、Android 与 iOS 等多端复用，且既拥有 Web 开发周期短、热更新等优势，也拥有原生系统交互性能好的优势。大大节省人力资源，缩短开发周期[3]，尽快将项目投入到运维中。

近年来，为了降低移动应用开发成本，提高开发效率[4]，跨平台开发框架层出不穷[5]。首先是以 PhoneGap、AppCan 为代表的 Hybrid App（混合模式移动应用）[6]，Hybrid App 只有一个 UI WebView，里面访问的是 Web App[7]。这种模式虽然在一定程度上实现了跨平台开发，但由于其访问的是 Web App，受不同平台浏览器内核响应速度的影响，存在访问速度慢、UI 反应延迟等问题[8]。

2016 年由 Facebook 开发的 React Nactive 框架以及阿里开发的 Weex 框架，打破了以往跨平台开发框架仅停留在 WebView 层面的不足，使用户仅使用 HTML5，JavaScript，CSS 即可开发 Web，Android、iOS 原生应用[9][10]。从此，跨平台开发开启了新的里程碑！

1.2 研究意义

本文通过使用 Weex 框架开发一个微阅读应用（包括 Android 端与 Web 端），深入学习 Weex 跨平台实现原理及运行流程。比较采用 Weex 跨平台开发与原生开发在交互性、UI 编写、版本更新迭代等方面的异同，体会 Weex 在跨平台开发时的优点与不足。通过对比 PhoneGap、React Native 与 Weex 等跨平台开发框架，给想要技术转型的同学以参考意义。同时，由于 Weex 框架在 2016 年初次应用到淘宝、天猫双十一会场，目前版本为 1.0.0，版本更新迭代频繁，尚存在很多亟待解决的问题，本文将总结在项目开发时遇到的问题并给出相应的解决方案，供学习 Weex 框架的同学参考。

1.3 本文章节安排

本文的章节安排如下：

第 1 章：绪论。主要介绍本论文的研究背景、研究意义以及本文的章节安排。

第 2 章：文献综述。跨平台开发已成为一种趋势，市面上跨平台开发解决方案众多，这一章就各种跨平台开发框架进行比较分析，并概述 Weex 框架的优点。

第 3 章：系统需求。本章主要介绍微阅读项目的具体功能。

第 4 章：项目概要设计。本章完成项目的结构设计与架构设计，根据系统需求将系统划分为三大功能模块并概述每个模块的设计。

第 5 章：项目详细设计。本章详细描述每个功能模块的设计。

第 6 章：系统实现。本章描述系统开发工具及运行环境，并画出函数流程图描述各模块的具体实现。且将项目集成到 Android，使其在 Web 与 Android 端都能运行。

第 7 章：测试。本章对各功能模块进行测试。

第 8 章：结论。本章总结论文所做工作。

2 文献综述

2.1 跨平台开发框架比较

2.1.1 PhoneGap

2008 年，PhoneGap（现已命名为 apache Cordova）的推出，拉开了跨平台开发的序幕[11]。PhoneGap 旨在不同的移动设备上都可以像运行本地应用一样运行 Web 应用[12]。其采用标准的 HTML、JavaScript、CSS 等 Web 技术开发，通过不同平台的内置浏览器加载 HTML 相关文件来实现跨平台[13]，降低开发难

度，受到了广大 Web 开发者的青睐。其具有明显的优势-----开发成本低，Web app 不用做任何修改即可复用到 Android、iOS 系统，但受浏览器处理速度的影响，使用 PhoneGap 构建的应用程序，存在运行速度慢，UI 反应延时等问题[14]。且有测试表明，运行 PhoneGap 应用内存消耗远大于原生态应用。对于频繁操作，释放内存较慢。

2.1.2 React Native

2015 年，Facebook 推出了基于 JavaScript 的跨平台开发框架 React Native（RN）。RN 的整个框架就是为了解释运行编译后的 js 脚本文件，通过判断 js 中的 UI 元素来调用原生系统的渲染机制绘制页面。RN 使开发人员仅使用 JavaScript 便可开发 Android、iOS 原生应用，打开了跨平台开发的新篇章。它不是运行在 WebView 之上的，所以彻底摆脱了 WebView 交互性能差的问题。React Native 一推出便受到了业界人士的关注，目前，RN 社区十分活跃，在 github 上有 4w+satr，Fork、Issues 也比较多[9]。但使用 RN 需学习 React 框架、全新的语法 jsx、ES6 语法和函数式编程思想，学习成本较高，且用 RN 开发出的应用只能跨 Android 与 iOS 两端，其并没有对 Web 端做兼容处理，不能实现代码的三端复用。

2.1.3 Weex

2016 年，阿里推出了 Weex 跨平台开发框架，使开发者仅使用 HTML，JavaScript，CSS 便可开发 Android、iOS 原生应用。Weex 整体架构如图 2-1，开发者可在本地编写 Weex 文件，并将其编译为 js 文件部署到服务器上，而在移动应用客户端里，Weexsdk 会准备一个 js 引擎去解析 js 文件，根据解析的结果，通过 bridge 调用原生系统中的相应的页面渲染机制、函数方法等。2016 年 11 月 11 日，应用在手淘、天猫上的 Weex，性能得到了很好的验证。Weex 遵循 Web 开发标准，学习成本较低，上手较快，且 Weex 的宗旨是“write once，run everywhere”，代码仅需编写一次，即可实现多端的复用，支持 Android、iOS、Web 平台。目前，阿里旗下产品淘宝、天猫、钉钉等均在使用 Weex 进行开发。

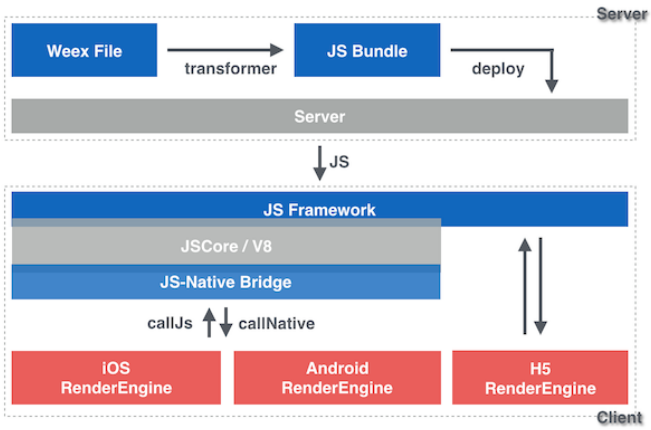

图 2-1 Weex 整体架构图

2.1.4 小结

从性能来看，使用 Weex 开发，最终调用的是原生系统的渲染机制与函数方法，提供给用户不输于原生应用的交互体验；

从学习成本来看，Weex 遵循 Web 开发标准，开发者仅需学习基础的 HTML 语言、CSS 样式、JavaScript 语法，即可着手进行开发，技术门槛极低；

从代码复用率来看，Weex 仅需少量适配，即可跨 Web、Android、iOS 三端，代码复用率极高。

综上，不管从性能、学习成本还是代码复用率来看，Weex 都是一款当下比较流行且值得去研究的框架。本项目选择使用 Weex 进行开发，进而对其做深入学习与研究。

2.2 使用 Weex 开发与原生开发的异同

2.2.1 UI 页面编写

原生开发中，一般公司会采取美工与开发人员协作开发的模式[15]，美工专注于 UI 页面的设计，而程序员专注 UI 页面的编写以及页面逻辑的实现。通常的做法是美工完成 UI 标注图，即:标注页面背景

颜色、组件尺寸大小与颜色、字体大小与颜色等，然后由开发人员编写页面布局。如：Android 中的 xml 文件，Web 中的 HTML 与 CSS。这种做法不仅会造成 Web、Android、iOS 系统上的 UI 页面无法完全统一，而且存在美工的设计与程序员所编写的 UI 无法完全一致的问题。在页面完成后，不管美工还是程序员，都面临再次修改的问题[16]。在 Android 开发中，本身 xml 布局较为繁琐，在对应 Activity 中又多次出现与代码逻辑连在一起的 findViewById()，修改起来非常麻烦。

引入 Weex 之后，Weex 遵循 Web 标准，且提供了 http://dotwe.org，一个支持实时预览的平台。公司可以让美工和程序员独立工作，美工专注于 UI 页面的设计与编写，程序员只专注于页面逻辑的开发。这样就不存在需要开发者多次修改页面的问题，从而提高工作效率，缩短项目开发周期，降低公司开发成本。

2.2.2 开发周期与成本

就研发人员而言，在原生开发中，往往需要三组团队（Android、iOS、Web）来完成一个应用的开发[17]。若一个团队有四个人，大概也需要 12 个人来完成前端三端。这就极大的增加了公司的成本。而使用 Weex，一套代码可实现三端的复用，开发人员可大量减少，降低公司成本。

就测试而言，在原生开发中，测试组人员往往需要分别测试三端的代码，而对于一个应用来说，其业务逻辑是一定的，所以测试人员往往会去校验三次业务逻辑[18]，工作效率可想而知。而使用 Weex，代码实现高度复用，虽然测试组人员依然要测试三端，但工作量却大大的减少了。

就开发周期而言，开发原生应用尤其是 Android 与 iOS，需要大量的时间与精力，完成一个项目往往需要四到六个月。而使用 Weex 开发，一个人的代码，可以在三端复用，大大缩短开发周期，使项目尽快投入到运维中。

2.2.3 版本更新迭代

一个应用完成之后，版本更新迭代是常见的现象。在原生开发中，对于开发者而言，进行版本更新时，往往伴随着代码混淆、打包，向各应用商店提出版本更新申请，等待更新。这一些列繁琐的操作，不仅影响版本迭代周期，而且影响开发效率。对于用户而言，每次的版本更新，都需要重新下载安装包，浪费流量且操作繁琐，导致部分用户不愿更新。

而使用 Weex 开发，对于开发者而言，由于是项目中的 WeexSdk 解析 js 文件，调用系统原生方法绘制组件。我们将 js 文件全部部署在云端，当需要更新时，直接替换掉相应的 js 文件。这样就节省了进行代码混淆、打包的时间，提高了项目开发效率。且不需要向各应用商店提出申请，不用等待审核结果，不存在所谓审核周期的概念。对于用户而言，不需要再次下载安装包，即可实现更新。

3 系统需求

3.1 项目概述

微阅读项目主要包括三部分，图片模块、阅读模块和电影模块。

图片模块，该模块主要向服务端获取某月份图片及文字数据，并进行展示。图片列表页向用户展示图片且每张图片搭配一句名人名言，用户可向左滑动图片显示上一张，向右滑动图片显示下一张；同时，用户可以通过点击标题栏的月份选择按钮，弹出月份选择框，选择某个月份之后，向网络请求该月份下的图片列表数据并进行展示。为了提高用户体验感，增强交互性，当用户点击图片后，会动态隐藏标题栏与图片上的文字，再次点击时，会动态显示标题栏与文字。

阅读模块，该模块主要向服务端获取一些精美文章数据，并进行展示。文章列表页主要向用户展示文章的创作时间、文章标题、作者及文章概述，由于文章数量比较多，当用户向下滑动时，会向服务端发送请求，加载更多数据进行显示。当用户点击某项文章之后，即可进入相应文章详情页，查看文章具体内容。

电影模块，该模块主要向用户展示一些电影的海报、导演信息、剧情梗概、关键字及评分等信息。当用户进入电影模块，即可看到一个长度为 10 的电影海报列表，并且每张海报右下角有该电影的评分。当用户点击某个电影海报之后，即可进入相应的电影详情页。在电影详情页，首先是 2-4 张电影的海报

在自动播放，然后显示该电影的关键字、导演信息、剧情梗概。

3.2 功能需求

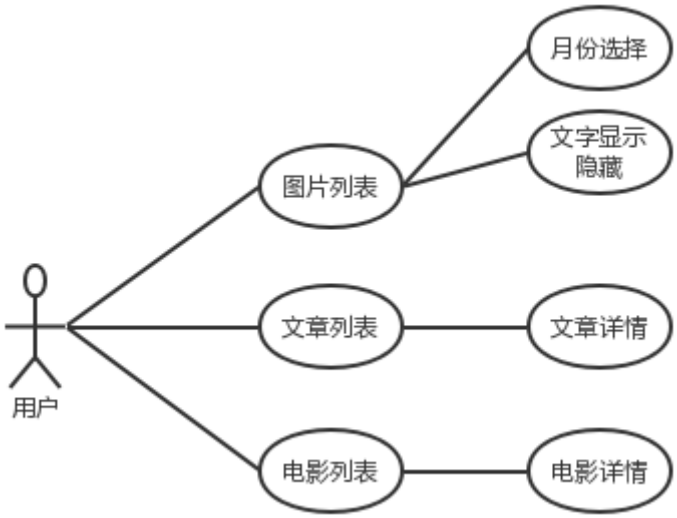

图 3-1 用例图

根据项目概述，确定微阅读应具有以下功能：

（1）用户通过点击底部导航栏按钮，进入图片列表页、文章列表页及电影列表页。

（2）用户在图片列表页，选择月份，进行不同月份图片及文字信息的切换。

（3）用户在图片列表页，点击某张图片，使得标题栏与文字信息显示与隐藏。

（4）用户在文章列表页，点击某一项文章，进入其文章详情页。

（5）用户在文章详情页，点击返回按钮，返回到文章列表页。

（6）用户在电影列表页，点击某一张电影海报，进入相应电影详情页。

（7）用户在电影详情页，点击返回按钮，返回到电影列表页。

4 项目概要设计

4.1 系统分析

本项目基于 Weex 进行开发，开发完成后，可直接在 Web 浏览器上运行；若要打包出可在 Android 手机上运行的 apk 文件，需要新建 Android 项目，并做相关集成。该项目最终实现代码在 Web 与 Android 端复用，且打包出可在 Web 与 Android 端运行的应用。

4.2 系统功能模块设计

如图 4-1，微阅读项目主要包含以下几个功能模块。

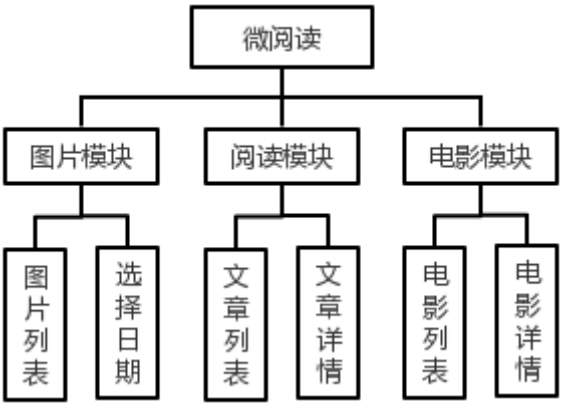

图 4-1 微阅读项目模块图

4.2.1 图片模块

该模块需要根据用户选择的月份向服务端请求图片及文字相关数据，接口文档设计如表 4-1 和表 4-2。整个模块主要涉及到 photo.we、photo-titlebar.we。photo-titlebar.we 主要封装图片列表页标题栏 UI 及月份选择按钮点击事件。在 photo.we 中使用 photo-titlebar.we，并对其月份选择按钮点击事件进行监听，同时，封装网络请求方法、点击图片处理事件、显示与隐藏月份选择框方法、显示与隐藏标题栏和文字信息等方法。

表 4-1 图片模块请求参数表

参数名称	数据类型	必选	描述	备注
month	String	true	请求的年月	如:2017-03

表 4-2 图片模块返回参数表

元素名称	数据类型	元素意义	备注
res	int	服务器处理结果	0 表示处理成功
data	array	图片数组	数组长度=当月天数
image_id	String	图片 id	方便后期查找图片信息
image_time	String	图片更新时间	格式:2017-03-27
image_title	String	图片的标题	无
image_content	String	图片相关内容	无
image_url	String	图片地址	无

4.2.2 阅读模块

该模块需要从服务端获取当天文章列表数据及文章详情数据，接口文档设计如表 4-3、表 4-4、表 4-5、表 4-6。整个模块主要涉及到 essay.we、essay-detail.we。essay.we 负责文章列表页的显示，其中封装网络请求方法、加载更多方法、进入文章详情页方法等。essay-detail.we 负责文章详情页的显示，其中封装网络请求方法，用于获取文章详情信息，且监听返回按钮点击事件，返回到文章列表页。

表 4-3 文章列表请求参数表

参数名称	数据类型	必选	描述	备注
index	String	true	请求的页码	从 0 开始

表 4-4 文章列表返回参数表

元素名称	数据类型	元素意义	备注
res	int	服务器处理结果	0 表示处理成功
data	array	文章数组	数组长度等于 9
essay_id	String	文章 id	方便后期查找文章信息
essay_title	String	文章标题	无
essay_time	String	文章发表时间	无
essay_author	String	文章作者	无
essay_info	String	文章概述	无

表 4-5 文章详情请求参数表

参数名称	数据类型	必选	描述	备注
essay_id	String	true	文章 id	无

表 4-6 文章详情返回参数表

元素名称	数据类型	元素意义	备注
res	int	服务器处理结果	0 表示处理成功
data	array	文章对象	无
essay_id	String	文章 id	方便后期查找文章信息
essay_title	String	文章标题	无
essay_time	String	文章发表时间	无
essay_author	String	文章作者	无
essay_info	String	文章概述	无
essay_author_url	String	作者头像地址	无
essay_content	String	文章内容	无

4.2.3 电影模块

此模块默认从服务端获取当天电影影评信息及电影详情信息，接口文档设计如表 4-7、表 4-8、表 4-9、表 4-10。整个模块主要涉及到 movie.we 和 movie-detail.we。movie.we 中封装了用于获取电影列表信息的网络请求方法和进入电影详情页的方法。movie-detail.we 中封装了网络请求方法，用于请求电影详情信息，同时监听返回按钮点击事件，返回到电影列表页。

表 4-7 电影列表请求参数表

参数名称	数据类型	必选	描述	备注
day	String	false	请求的日期	如:2017-03-27

表 4-8 电影列表返回参数表

元素名称	数据类型	元素意义	备注
res	int	服务器处理结果	0 表示处理成功
data	array	电影数组	数组长度等于 10
movie_id	String	电影 id	方便后期查找电影信息
movie_title	String	电影标题	无
movie_score	String	电影评分	无
movie_cover_url	String	电影海报图片地址	无

表 4-9 电影详情请求参数表

参数名称	数据类型	必选	描述	备注
movie_id	String	true	电影 id	无

表 4-10 电影详情返回参数表

元素名称	数据类型	元素意义	备注
res	int	服务器处理结果	0 表示处理成功
data	对象	电影对象	无
movie_id	String	电影 id	方便后期查找电影信息
movie_title	String	电影标题	无
movie_score	String	电影评分	无
movie_keywords	String	电影关键字	无
movie_cover_url	String	电影海报图片地址	无
movie_photo_list	array	电影图片地址集合	无
movie_base_info	String	导演编剧等信息	无

movie_info	String	电影剧情梗概	无

4.3 项目结构设计

Weex 项目主要使用 HTML、CSS、JavaScript 进行开发，开发完成之后，其可直接运行在浏览器中；在 Android 或 iOS 平台主要是加载其编译后的 js 文件。Weex 项目结构设计如图 4-2。

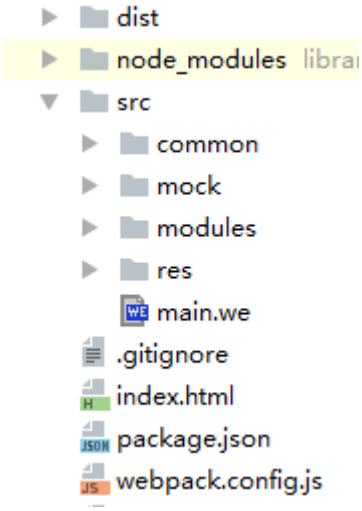

图 4-2 Weex 项目结构设计图

（1）dist：该文件夹用来存放项目编译后生成的 js 文件，在 Android 项目中，可直接将该文件夹放到 assets 文件夹下，进行本地 js 加载。

（2）node_modules：存放项目中用到的相关依赖。

（3）common：该文件夹主要用来存放项目中会在多个地方用到的类，内含 api.js 程序，其 getData 函数，封装了网络请求方法，其 getBaseUrl 函数，封装了判断当前运行环境（Android、iOS、Web）的方法，供其他模块调用。

（4）modules：该文件夹下存放 we 文件，可编译为 js 文件，在 Android 与 iOS 中被加载。

（5）res：该文件夹下用来存放图片、图标等资源文件。

（6）index.HTML：该程序是运行在浏览器上的入口程序，可在其中配置初次加载的 we 文件。

（7）package.json：项目的配置文件，配置相关依赖。

（8）webpack.config.js：项目打包的配置文件。

4.4 项目架构设计

Weex 项目程序架构图如图 4-3，在 index.we 程序中，使用 wxc-tabbar（底部导航栏），配置 item 为 photo、read、movie，并设置其对应 js 路径。对于 essay、movie 模块，均可点击某一项查看具体详情，需为其设置相应的点击函数，并传入对应的 uri 路径。

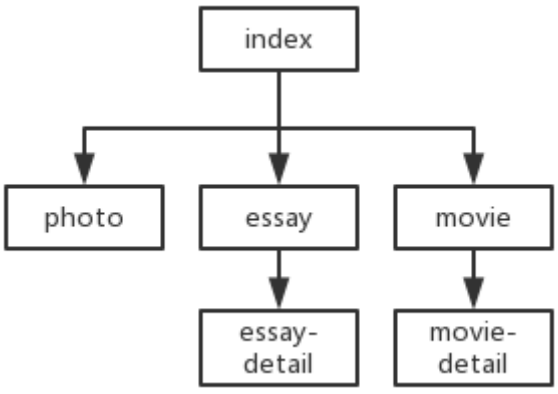

图 4-3 Weex 项目程序架构图

5 项目详细设计

5.1 通用模块

5.1.1 通用方法

在 common 文件夹下的 api.js 中，封装以下函数，被其他模块调用。此部分具体代码可参考附录中的 api.js。

（1）function getData(url, callback) {}

该函数用于网络请求，其内部调用 stream.sendHttp()进行网络请求，参数 url 表示请求链接，callback 用于数据的返回。在图片模块、阅读模块和电影模块进行网络请求时，最终调用的是它。

（2）exports.getBaseUrl = function (bundleUrl){}

该函数用于判断当前应用运行的环境，并根据不同的环境，返回相应的 baseUrl。如：其通过判断 bundleUrl.indexOf('file://assets/')是否大于等于 0，来判断是否是 Android 环境。在图片模块、阅读模块和电影模块，我们使用 this.$getConfig().bundleUrl 来获取当前应用运行环境的 bundleUrl，并将其作为参数传入 api.getBaseUrl()中。

5.1.2 通用 UI

项目中页面上的标题栏样式是一样的，将其封装为 base-titlebar.we。设置 title 与 showBack 属性，根据使用 base-titlebar 的页面所传入的参数，来确定 titlebar 中间显示文字以及左边返回键是否显示。

5.2 图片模块

5.2.1 UI 实现

该模块使用 slider 组件实现图片左右滑动的效果，为了兼容不同尺寸的手机，在 slider 组件外包裹 scroller 组件，使图片可以上下滑动。使用 list、cell、text、div 等组件实现月份选择框。图片模块最终效果如图 4-1 和图 4-2。

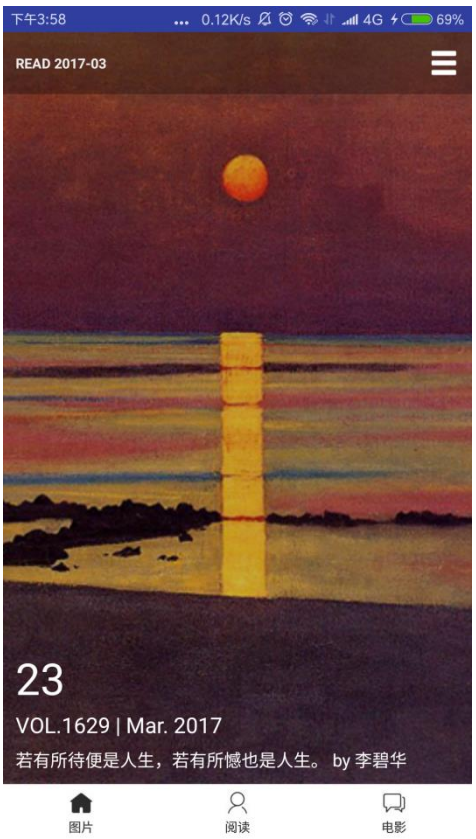

图 4-1 图片模块显示图

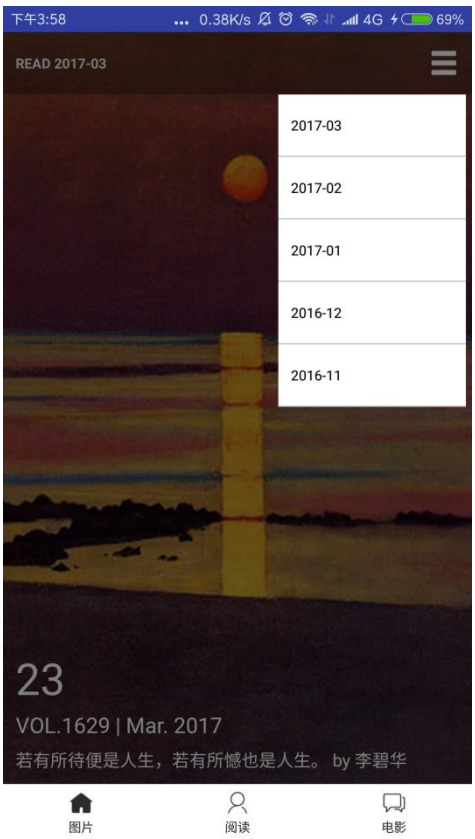

图 4-2 图片模块日期选择图

5.2.2 主要函数设计

此部分主要代码可参考附录中 photo.we。

（1）getPhotoList: function(String dateStr){}

该函数用于向服务端请求图片列表数据，dateStr 表示请求的年月，如’2017-03’。

（2）showMenu: function(){}

该函数在点击标题栏选择月份按钮后被调用，用于弹出日期选择框。

（3）hideMenu: function(){}

该函数在点击空白处或选择月份之后被调用，用于隐藏日期选择框。

（4）onMenuItemclick: function(){}

该函数在选择日期后调用，在其中依次执行 getPhotoList()、hideMenu()，向服务端请求所选年月的图片列表数据并进行展示，隐藏日期选择框。

（5）onImageClick: function(){}

该函数在点击首页图片时被调用，用于隐藏标题栏及文字信息。

5.3 阅读模块

5.3.1 UI 实现

（1）阅读列表页

该页面中引入 base-titlebar，title 属性设置为“阅读”，showBack 属性设置为 false，不显示返回按钮。使用 div 与 text 组合实现 essay-item，在 list cell 中引入 essay-item，并添加 loading 组件实现向下滑动加载更多。最终效果如图 4-3。

（2）阅读详情页

该页面中引入 base-titlebar，title 属性设置为“阅读”，showBack 属性设置为 true，显示返回按钮。使用 scroller、div、image、text 等组件完成页面布局。最终效果如图 4-4。

图 4-3 图片列表页

图 4-4 文章列表页

5.3.2 主要函数设计

此部分主要代码可参考附录中 essay.we。

（1）getEssayList: function(int index){}

该函数用于向服务端请求文章列表数据，参数表示请求的页码。

（2）onLoading: function(int index){}

该函数用于下拉加载更多，在其中执行 index++，并再次向服务端请求数据。

（3）goEssayDetail: function(int index){}

该函数在点击文章列表某一项时后被调用，通过调用 navigator.push 方法，跳转到文章详情页，参数 index 标识详情页 url。

（4）goBack: function(int index){}

该函数在点击标题栏上的返回按钮后被调用，用于从文章详情页返回到文章列表页。

5.4 电影模块

5.4.1 UI 实现

（1）电影列表页

该页面中引入 base-titlebar，title 属性设置为“电影”，showBack 属性设置为 false，不显示返回按钮。使用 list、div、image、text 等组件完成页面布局，最终效果如图 4-5。

（2）电影详情页

该页面中引入 base-titlebar，title 属性设置为“电影”，showBack 属性设置为 true，显示返回按钮。使用 slider 与 image 组合实现电影海报的轮播，使用 scroller、div、text 等组件完成整体布局。最终效果如图 4-6。

图 4-5 电影列表页

图 4-6 电影详情页

5.4.2 主要函数设计

此部分主要代码可参考附录中 movie.we。

（1）getMovieList: function(index){}

该函数用于向服务端请求电影列表数据，无参数，服务端会返回一个长度为 10 的数组，里面是 movie 对象。

（2）onMovieItemClick: function(index){}

该函数在点击电影列表某一项后被调用，通过调用 navigator.push 方法，跳转到电影详情页。

（3）goBack: function(index){}

该函数在点击标题栏上的返回按钮后被调用，用于从电影详情页返回到电影列表页。

6 系统实现

6.1 开发工具

操作系统：Windows 10

语言：HTML、CSS、JavaScript、Java

工具：WebStorm、Android Studio

6.2 运行环境

Web 浏览器、Android 智能手机

6.3 各模块具体实现

6.3.1 图片模块

该模块实现如图 6-1，用户进入图片模块后，首先调用 getPhotoList()请求当月图片列表，并进行展示；当用户点击月份选择按钮后，调用 showMenu()，弹出月份选择框，用户选择月份后，调用 onMenuItemclick()，进行点击事件的处理，会再次调用 getPhotoList()，向服务端发送请求并解析返回的数据进行显示。

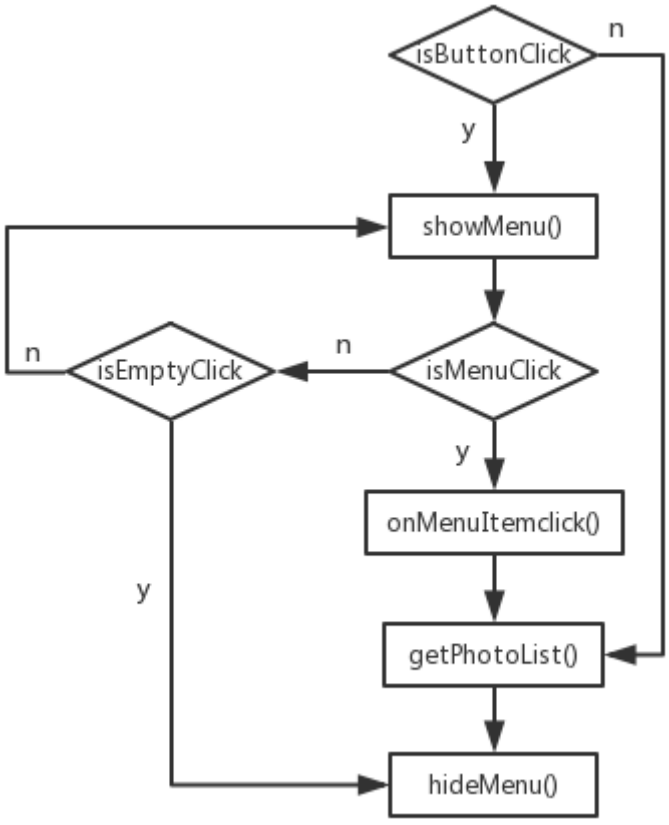

图 6-1 图片模块实现图

6.3.2 阅读模块

该模块实现如图 6-2，进入阅读模块后，首先调用 getEssayList()，传入参数 index=0，请求第一页文章列表；当用户下滑至第一个页面数据展示完全时，会执行 index++，再次调用 getEssayList()，请求第二页文章列表数据。当用户点击文章列表中的某一项，会调用 goEssayDetail()，打开文章详情页。

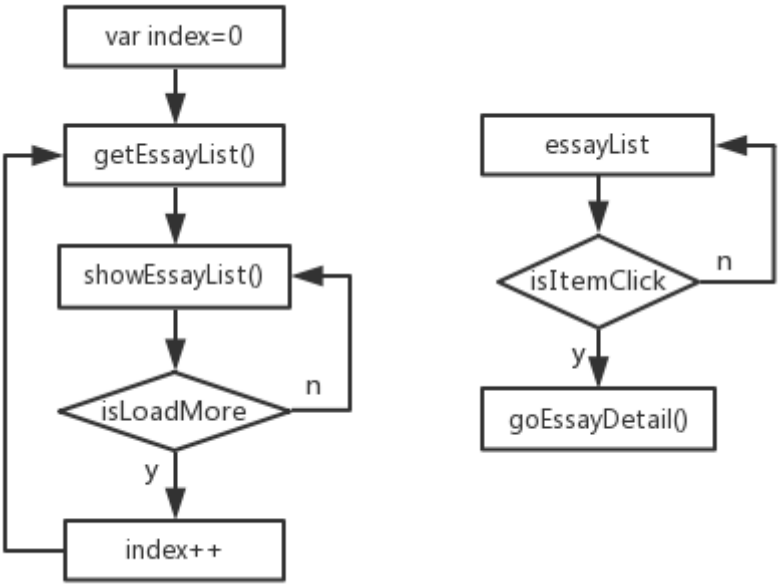

图 6-2 阅读模块实现图

6.3.3 电影模块

该模块实现如图 6-3，用户进入电影模块后，调用 getMovieList()，进行网络数据请求，根据返回的结果调用 showMovieList()，进行图片列表的展示。当用户点击某一项图片，将调用 goMovieDetail()进入电影详情页。

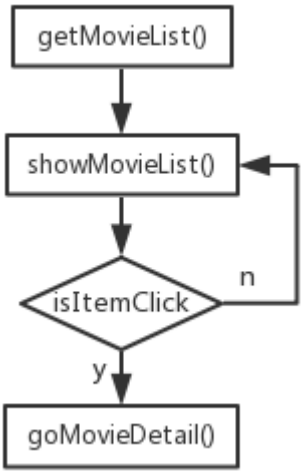

图 6-3 电影模块实现图

6.4 集成到 Android

使用 Weex 开发，可以使用 Weexpack 新建 Weex 项目，其中包括 Web、Android 与 iOS 三端；也可以使用 weex-toolkit 建立项目，然后，将开发所得 js bundle，集成到 Android 与 iOS。本项目使用的是 weex-toolkit，集成主要包括两大部分：初始化与页面渲染。

6.4.1 初始化

如图 6-4，在 WXApplication 的 onCreate()中，将 ImageAdapter 实例作为参数传给 WeexSdk 的初始化函数 initialize，然后调用 registerModule()注册自定义事件。

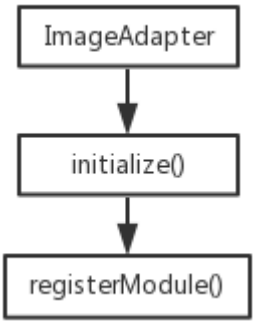

图 6-4 Android 端初始化实现图

6.4.2 页面渲染

如图 6-5，首先调用 WXSDKInstance 构造函数初始化 WXSDKInstance 实例，然后调用 registerRenderListener()注册页面渲染监听，renderByUrl()用来加载对应 uri，并进行渲染，preRenderPage()与 postRenderPage()，用来处理在页面渲染之前、页面渲染消息发出之后，页面上布局的变化。若页面渲染成功，将进入 onRenderSuccess()，否则，进入 onException()。

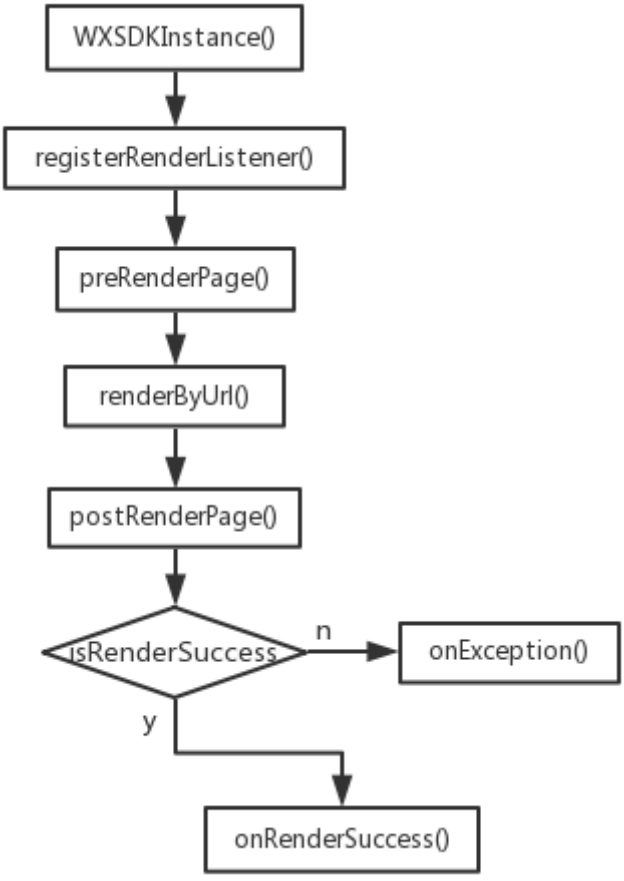

图 6-5 Android 端页面渲染图

6.5 遇到的问题与解决方案

6.5.1 无法加载 so 包

出现这个问题的原因是 WeexSdk 支持的 abi 种类不够多，解决方法是在 Android 项目 build.gradle 中，添加 ndk{ abiFilters "x86" abiFilters "armeabi"}，限制 apk 使用的 abi 种类。

6.5.2 获取不到 Android 运行环境

出现这个问题的原因是在 Android 项目中没有将标识 Android 端的字符串传到 renderByUrl 中。解

决方法是将标识 Android 端的字符串存入 map 中，作为 renderByUrl 函数的第三个参数。

7 测试

7.1 测试方法

一部分同学通过将打包出的 apk 装在 Android 手机上体验该应用，一部分同学通过访问 Web 地址来体验该应用。

7.2 测试环境

Android 智能手机与 Web 浏览器

7.3 测试对象

主页、图片列表页、文章列表页、文章详情页、电影列表页、电影详情页。

7.4 测试用例

7.4.1 主页测试用例

表 7-1 主页测试用例

测试序号	前置条件	实际操作	预期结果	测试结果
1	进入主页	点击图片按钮	显示图片列表页	通过
2	进入主页	点击阅读按钮	显示文章列表页	通过
3	进入主页	点击电影按钮	显示电影列表页	通过

7.4.2 图片模块测试用例

表 7-2 图片模块测试用例

测试序号	前置条件	实际操作	预期结果	测试结果
1	进入图片界面	点击选择月份按钮	弹出月份选择框	通过
2	进入图片界面	选择月份	请求图片列表	通过
3	进入图片界面	选择月份	隐藏月份选择框	通过
4	进入图片界面	向右滑动图片	显示下一张图片	通过
5	进入图片界面	向左滑动图片	显示上一张图片	通过

7.4.3 阅读模块测试用例

表 7-3 阅读模块测试用例

测试序号	前置条件	实际操作	预期结果	测试结果
1	进入文章列表页	向下滑动页面	加载更多数据	通过
2	进入文章列表页	选中某一项	进入文章详情页	通过
3	进入文章详情页	点击返回按钮	回到文章列表页	通过

7.4.4 电影模块测试用例

表 7-4 电影模块测试用例

测试序号	前置条件	实际操作	预期结果	测试结果
1	进入电影列表页	选中某一项	进入电影详情页	通过
2	进入电影详情页	点击返回按钮	回到电影列表页	通过

7.5 测试结论

所有测试结果与预期相同。

8 结论

8.1 论文工作总结

本文通过使用 Weex 框架开发一款微阅读应用，实现了仅在 Android 端做少量适配，即可一套代码复用 Android 与 Web 两端，大大节省了开发周期且实现了热更新。通过对 Weex 的学习与实践，个人觉得不论是从学习成本还是开发效率来看，其无疑是当下的一个技术亮点。

开发过程中，使用最多的是 HTML、css、JavaScript 语言，只有集成到 Android 时，才使用了 Java 语言。由于对 JavaScript 语法不是很熟练，一开始遇到了很多问题，不过这些都通过后来的学习得到了很好的解决。

如果有更多时间的话，将对项目中使用的网络请求方式做进一步地优化与尝试，现在项目中网络请求是在 Weex 项目中完成的，可以进一步尝试在 Android 端进行网络请求。

如果可以再做一次该项目的话，将对集成到 Android 做进一步修改，将现在项目中使用的 sdk 依赖改为源码依赖，使用源码依赖方便进一步查看 WeexSdk 源码，且可通过修改源码进行调试，进一步学习与研究。

通过这次毕业设计，对混合开发、跨平台开发概念有了新的认识与理解，对项目结构设计、项目架构设计有了新的体会，且解决问题的能力得到了锻炼与提高。

8.2 问题与展望

Weex 跨平台开发框架 2016 年底推出，目前网络上的教程还不是很多，且在性能优化、兼容 Web、iOS、android 三端等方面还存在一些问题，需要开发者自己修改源码来实现其需求。但个人认为这些问题都会随着时间被解决，其流行是不可避免的，跟随 Weex 一起成长，将会学到很多！

致谢

首先，我要感谢我的指导老师张金颖老师，从论文的选题、撰写到最终的定稿，张金颖老师都给了我悉心的指导与热情的帮助，使我的毕业论文能够顺利完成。张金颖老师对工作的认真负责、对学术的钻研精神和严谨的学风，都是值得我终生学习的。

其次，感谢软件学院的全体领导和老师，在他们的悉心教导下，我掌握了扎实的专业技能。毕业在即，在今后的工作和生活中，我会铭记师长们的教诲，继续不懈努力和追求！

最后，感谢我的家人在此期间给予我的包容、关爱和鼓励，以及所有陪我一路走来的同学和朋友，正是由于他们的支持和照顾，我才能安心学习，并顺利完成我的学业。

参考文献

[1] 张露.移动多平台跨平台开发工具集的设计与实现[硕士学位论文].武汉:华中科技大学,2013,1-18.

[2] Charland A,Leroux B.Mobile application development:Web vs.native.Communications of the ACM, 2011, 54(5): 49–53.

[3]Song Sun,San Xing Cao.The Web Development Technology Research of Cross Platform Mobile Application[J]. Applied Mechanics and Materials,2014,3468(644):3090-3093.

[4] 孟凡宁,丛中昌,黄志兴,贺楚瑜. 移动互联网应用跨平台开发研究[J]. 移动通信,2013,(13):60-63.

[5] Suyesh Amatya, Arianit Kurti. Cross-Platform Mobile Development: Challenges and Opportunities. [J]. ICT Innovations and Education, 2014, 219-229.

[6] Hartmann G, Stead G, DeGani A. Cross-platform mobile development.Mobile Learning Environment, Cambridge,2011,5-26.

[7] Xian Wei Li. Research of Android Platform Application Based on HTML5[J]. Applied Mechanics and Materials,2014,3512(651):1901-1904.
[8] 黄永慧,陈程凯. HTML5 在移动应用开发上的应用前景[J]. 计算机技术与发展,2013,(07):207-210.
[9] https://github.com/facebook/react-native.
[10] https://weex.apache.org.
[11] 戴炳荣,王宇飞,王凌,陆奇峰,徐承宇. 基于 PhoneGap 跨平台移动开发框架的研究与应用[J]. 现代计算机(专业版),2016,(27):30-35.
[12] Brooks, Jason. PhoneGap 1.3 Adds Windows Phone 7 Support[J]. EN,2012,29(2):17-18.
[13] 崔丽梅,薛斐斐.基于 PhoneGap 技术的跨平台移动应用开发探讨[J]. 电脑知识与技术,2016,(15):30-33.
[14] 徐隆龙,李莹,白静.移动混合开发框架[J].计算机系统应用,2014,(12):53-59.
[15] 郑秋梅,刘真,苏政,李松. 基于 Android 的自定义通用可视化控件[J]. 计算机系统应用,2017,(01):222-226.
[16] Lin S F, Zhou Y, Wang R Y, et al. GoogleMap Application Development in Android Platform[J]. Applied Mechanics & Materials, 2014, 513-517:466-469.
[17] 马友忠,孟小峰,姜大昕.移动应用集成:框架、技术与挑战[J].计算机学报,2013,36(7):1375-1387.
[18] 胡忠望, 莫家庆. 基于 Android 的移动应用开发研究[J]. 软件导刊, 2014(11):59-78.

Study and Practice of System Development Based on Weex Cross-Platform

---- Illustrated by the Case of APP < Micro-Reading >

123022013062 Liu Ting Supervisor:Zhang Jinying
Faculty of Software, Fujian Normal University, Fuzhou350108

Abstract: Based on Weex cross-platform development framework, developers can develop Android and iOS native applications only use HTML, CSS and JavaScript , which opened a new milestone in cross-platform development. Micro-Reading is an application based on Weex cross-platform development framework, using webstorm and android studio, can be run on the Web and Android platform. The similarities and differences between Weex and the native development are analyzed from the perspective of software development. The requirements analysis, project structure design, project architecture design and project realization are described in detail. Finally, the system is tested. The results show that codes based on Weex can be successfully written once, run everywhere.

Key words: Cross-Platform Development, Weex Framwork, Android

附录

api.js

```
function getData(url, callback) {
    stream.sendHttp({
        method: 'GET',
        url: url
    }, function (ret) {
        var retdata = JSON.parse(ret);
        callback(retdata);
    });
}

exports.getBaseUrl = function (bundleUrl, isnav) {
    bundleUrl = new String(bundleUrl);
    var nativeBase;
    var isAndroidAssets = bundleUrl.indexOf('file://assets/') >= 0;
    if (isAndroidAssets) {
        nativeBase = 'file://assets/dist/';
    }
    ......
    return nativeBase;
};
```

photo.we

```
getPhotoList: function (dateStr) {
    var self = this;
    self.showMainSlider = false;
    self.imageList = [];
    apis.getHome(dateStr, function (retdata) {
        var tempData = retdata.data
        self.nav_title = dateStr;
        for (var i = 0; i < tempData.length; i++) {
            var datap = tempData[i];
            var hpMarketTime = new Date(datap.hp_makettime.split(' ')[0]);
            datap.hp_day = hpMarketTime.getDate();
            datap.hp_month = hpMarketTime.getMonth();
            datap.hp_year = hpMarketTime.getFullYear();
        }
        self.showMainSlider = true;
        self.imageList = tempData;
    });
},
```

```
hideMenu: function (e) {
    var self = this;
    this.anim({
        opacity: 0
    }, this.$el('menuDiv'), 'ease-out', 200, function () {
        self.showMenu = false;
    });
},
onMenuItemclick: function (e) {
    this.$call('modal', 'toast', {
        'message': 'row ' + e.target.attr.rowdata + ' clicked',
        'duration': 2.0
    });
    this.getPhotoList(e.target.attr.rowdata);
    this.hideMenu(e);
}
```

essay.we

```
getEassyList: function (index) {
    var self = this;
    apis.getReadingIndex(index, function (ret) {
        if (!loadmore) {
            self.readingList = [];
        }
        self.showListItem = ret.data.length > 0;
    });
},
goEssayDetail: function (id) {
    var vm = this;
    var params = {
        'url': this.baseURL + 'modules/essay-detail.js?id=' + id,
        'animated': 'true'
    };
    navigator.push(params, function (e) {
        //callback
    });
},
onLoading: function (e) {
    var self = this;
    this.loading_display = "show";
    self.getEssayList(++self.currentIndex, true);
    self.loading_display = "hide";
},
```

movie.we

```
getMovieList:function(index){
    movie.getMovieList(self.index, function (ret) {
    self.movieItems = ret.data;
    });
}
onMovieItemClick: function (e) {
    var self = this;
    var itemid = e.target.attr.movieid;
    var params = {
        'url': this.baseURL + 'modules/movie-detail.js?id=' + itemid,
        'animated': 'true'
    };
    navigator.push(params, function (e) {
        //callback
    });
}
```