摘要

近几年来，智能手机的迅猛发展给人们的生活带来了极大的改变，移动互联网时代的来临使得人手拥有一台配置强悍的智能手机已不再是奢望。但随着社会老龄化问题的日益严重，智能手机市场对老年人市场的投入却依旧跟不上时代发展的步伐。老年人由于视力衰退，阅读报纸上的文字较为吃力，长时间戴眼镜更会加剧眼睛的疲劳状况。本设计提出了用智能手机帮助老年人阅读报纸的可能，并实现了一款基于 Android 智能手机的应用。

本设计实现了一款基于安卓智能手机的报纸阅读器，具有将拍照得到的图片转成文字再朗读出来的功能，并支持报纸内容存储、音量调节、自定义朗读语言等功能，能极大缓解老年人阅读报纸时的不便。论文首先介绍了 Android 开发平台以及 Cordova 跨平台开发方案，然后详细分析了该应用的所有功能需求，再结合应用开发的基础知识，讲解了开发所需的知识点，最后给了详细的代码实现方案。

关键词：Android；Cordova；报纸阅读器；OCR；TTS

In recent years, the rapid development of smart phones changed people's lives dramatically, and the advent of the mobile Internet era make everybody having a powerful configuration smartphone is no longer a luxury. But with an aging society is becoming increasingly serious, smart phone market for the elderly still can not keep up the pace of their development. Elderly due to vision loss, read the newspaper is more difficult to them, and long wear glasses can exacerbate eye fatigue condition. This design made smart phones may help the elderly to read the newspaper, and implemented of an Android-based smartphone application for them.

This design implemented an Android-based smartphone newspaper reader, which can transform photo into text and then read out it, and also support the content storage, volume control, custom speaking language to greatly alleviate the elderly inconvenience when reading the newspaper. This paper introduces the Android development platform and Cordova first, and then analysis of the the application of all functional requirements, combined with development and application of basic knowledge and explain the knowledge required to develop, finally give the detailed code scheme.

1. 引言

课题背景：

本课题为基于智能手机的报纸阅读器。老年人大多有读书看报的喜好，但受限于视力衰退，阅读报纸上较小的文字极为吃力，时间一长容易导致眼睛不舒服等不良状况，给日常生活造成了不小的麻烦。如果有人或者使用软件能够帮助老人将报纸内容朗读出来，这个问题就可以在一定程度上得到解决。

目前在 Android 应用市场有不计其数的 APP 应用，但提供给老年人使用的却少之又少，而且普遍质量较差，并没有针对老年人的特点和使用场景进行有针对性的设计和开发。结果就是这些 APP 并没有真正解决老年人的某些痛点问题。

本课题选择Android平台作为开发平台，因其使用范围广，基数大，可使其适用性大大增加。老年人只需使用该 App 对所需要阅读的内容拍照，本应用即会将选中的内容朗读出来，解决了老年人由于新闻正文字体较小而造成的阅读不便的问题。本 App 还提供了存储功能，用户可选择将之前阅读的内容缓存起来，便于下次使用。

课题任务：

本文的主要任务是介绍 Android 平台下使用 Cordova 进行的报纸阅读器应用开发。首先从整体的角度介绍了 Android 的整体架构，以及 Cordova 跨平台开发框架。然后对 Android 平台的报纸阅读器进行需求分析，设计应用的功能与业务流程。接着使用 Cordova 框架进行功能设计，描述具体实现过程。本论文旨在讲解使用 Cordava 框架进行 Android 平台应用开发的要点与过程，涉及到服务器端的功能设计，在此不会作详细的介绍。

本课题的整个设计开发过程均由本人独立完成。主要的工作包括下面几点：

1. 理解课题任务，学习应用开发所用到的开发技能，配置开发与测试环境。
2. 分析课题需求，设计业务流程、应用界面和代码实现。
3. 测试应用，解决测试时发现的问题，总结并完成本篇论文。

内容安排：

本论文通过六个章节来叙述课题概况。

第一章为引言，介绍课题的背景及意义，课题任务以及论文内容的安排。

第二章为开发平台的简介，介绍了 Android 平台，Cordova 跨平台开发框架。

第三章对课题进行需求分析，介绍了老年人对报纸阅读器的功能需求，以及本应用的操作流程分析。

第四章详细介绍了本应用的各项功能，以及对业务处理流程的分析。

第五章为详细的代码实现，介绍了各项功能的界面、数据处理等各方面细节。

第六章回顾了整个开发工作的过程以及对课题未来走向的展望。

1. 开发环境及相关技术简介

本章开始介绍开发所用到的环境和相关技术背景知识，以及选用这些技术方案的优势所在。主要介绍操作系统、开发语言、开发框架这三个构成基本课题开发的要素。

Android 简介

Android 是一个以 Linux 为基础的开放源代码的移动设备操作系统，中文俗称“安卓”，主要用于智能手机和平板电脑，由 Google 成立的开放手持设备联盟持续领导与开发中。目前 Android 已发布的最新版本为 Android 5.1.1。

Android 系统最初由安迪·鲁宾等人开发，最初开发这个系统的目的是创建一个基于数码相机的先进操作系统；但后来发现市场需求不大，加上智能手机市场快速成长，于是 Android 被改造成一款面向智能手机的操作系统。于2005年8月被 Google 收购。2007年11月，Google 与84家硬件制造商、软件开发商以及电信运营商成立了开放手持设备联盟来共同研发和改良 Android 系统。随后，Google 以 Apache 免费开放源代码许可证的授权方式发布了 Android 系统的源代码，让生产商推出搭载 Android 系统的智能手机与平板电脑。在2014年 Google I/O 开发者大会上 Google 宣布过去30天里有10亿台设备被激活。

Android 操作系统的核心属于 Linux 内核的一个分支，目前最新版的是基于 Linux 3.4.0 版的内核。Android的系统架构和其操作系统一样，采用了分层的架构。Android分为四个层，从高层到低层分别是应用程序层、应用程序框架层、系统运行库层和Linux内核层。在应用程序层，Android会同一系列核心应用程序包一起发布，该应用程序包包括客户端，SMS短消息程序，日历，地图，浏览器，联系人管理程序等。所有的应用程序都是使用Java语言编写的。应用程序框架层包括一系列的服务：视图、内容提供器、资源管理器、通知管理器、活动管理器等。系统运行库层包括一些 C/C++ 库，这些库能被Android系统中不同的组件使用。它们通过 Android 应用程序框架为开发者提供服务。核心库包括系统 C 库、媒体库、Surface Manager和LibWebCore。

Android应用程序是采用Java语言开发，传统的Java语言运行是通过JVM将源文件编译成Java字节码运行，但是Android虽然是Java语言开发，但是字节码并不是最终的运行格式，并且由于版本问题和移动设备电量等问题，Google自行开发了一个虚拟机运行Android程序，这个虚拟机称为Dalvik。当完成源代码的开发后，首先会将Java源码转换为字节码，然后通过DX工具，将字节码文件转换了.dex的文件，最后该dex格式的文件在Dalvik上运行。

Cordova 简介

Cordova 是一款开放源代码的跨平台开发框架，旨在让开发者使用 HTML、JavsScript、CSS 等 Web API 技术来开发跨平台的 APP，即将使用HTML、CSS、JavaScript的页面封装为手机客户端应用。原本由 Nitobi 公司开发，2011年10月4月，Adobe 正式宣布收购 Nitobi 软件。Cordova 是一个 APP 设备的 API 接口集，在页面与本地设备间提供了一座桥梁，允许开发者利用 JavaScript 调用如摄像头、罗盘等硬件资源，这是一般的 Web APP所做不到的。配合一些基于 HTML5、CSS3 技术的 UI 框架，如 jQuery Mobile，开发者可以快速地开发跨平台 APP 而不需要编写任何原生代码，实现了“一次开发，到处使用”，在很大程度上降低了移动开发的门槛。此外 Cordova 还拥有丰富的插件资源，很方便的供开发者进行功能拓展。Cordova 目前支持的操作系统包括苹果的 iOS、Google 的Android、Blackberry、LG 的 WebOS、微软的 Windows Phone、三星的 Tizen和 baba、Firefox OS 和Ubuntu Touch。

Cordova 能够实现跨平台开发的原因在于移动设备上都内置了 WebView 组件，这个内置的浏览器特性是 Web 能被打包成本地客户端的基础，可以方便的使用 HTML5、CSS3 等技术进行页面布局，在开发效率上优于使用原生开发。Cordova 针对不同平台的 WebView 组件做了拓展与封装，使得 WebView 组件可以访问设备本地的 API，所以开发者在 Cordova 框架下可以通过 JavaScript 访问到设备本地的 API。

不过由于使用 Web 技术，而且 Cordova 是一个封装了很多底层细节的框架，在运行效率上要比原生的应用程序低，在频繁操作时容易造成卡顿，另外在稳定性与内存占用上也不及原生应用。还未能做到100%支持原生接口，一些功能可能会因此受到限制，有待逐步在各平台完善。因此比较适合于开发功能简单以及操作不太复杂的 APP。

Cordova 原理简要分析

Cordova在其支持的平台上均实现了一套后台运行的框架，分别与各平台系统的API进行交互，从而调用其原生API，为开发者提供了统一的JavaScript接口。

Cordova是基于移动终端上内嵌的浏览器即WebView来渲染HTML、CSS，在Cordova中重写了WebView的onJsPrompt 方法，来捕获前端的业务请求操作，从而将Web前端与后台的Java程序连接起来。相应的，Android上的Cordova中使用Java实现了一个HTTP服务器，通过持久性的XHR连接，JavaScript可以不断轮询内部XHR服务器存储的信息，从而实现了从Java到JavaScript方向的通信。CallbackServer线程在Cordova框架被首次加载时就启动，然后会同时监听前端的XHR请求和链表中有无数据，当有XHR请求来时，若链表为空，线程会最多等待10s，在等待过程如果链表中有数据来，会唤醒等待的线程，从而立即将链表中数据传到前端；若10s后没有数据，则将空数据返回到前端，从而发起下一轮XHR请求。这就是“基于服务器推技术”的模型。

OCR简介

中文全称“光学字符识别”，是指是指电子设备（例如扫描仪或数码相机）检查纸上打印的字符，通过检测暗、亮的模式确定其形状，然后用字符识别方法将形状翻译成计算机文字的过程；即，针对印刷体字符，采用光学的方式将纸质文档中的文字转换成为黑白点阵的图像文件，并通过识别软件将图像中的文字转换成文本格式，供文字处理软件进一步编辑加工的技术。在1996年清华紫光开始搭配中文识别软件之前，市场上的扫描仪和OCR软件一直是分开销售的。扫描仪厂商现在已把专业的OCR软件搭配自己生产的扫描仪出售。OCR技术的迅速发展与扫描仪的广泛使用是密不可分的。近两年随着扫描仪逐渐普及和OCR技术的日臻完善，OCR己成为绝大多数扫描仪用户的得力助手。

光学文字识别的概念是在1929年由德国科学家Tausheck最先提出来的,后来美国科学家Handel也提出了利用技术对文字进行识别的想法。而最早对印刷体汉字识别进行研究的是IBM公司的Casey和Nagy,1966年他们发表了第一篇关于汉字识别的文章,采用了模板匹配法识别了1000个印刷体汉字。

早在60、70年代,世界各国就开始有OCR的研究,而研究的初期,多以文字的识别方法研究为主,且识别的文字仅为0至9的数字。以同样拥有方块文字的日本为例,1960年左右开始研究OCR的基本识别理论,初期以数字为对象,直至1965至1970年之间开始有一些简单的产品,如印刷文字的邮政编码识别系统,识别邮件上的邮政编码,帮助邮局作区域分信的作业;也因此至今邮政编码一直是各国所倡导的地址书写方式。

20世纪70年代初,日本的学者开始研究汉字识别,并做了大量的工作。中国在OCR技术方面的研究工作起步较晚,在70年代才开始对数字、英文字母及符号的识别进行研究,70年代末开始进行汉字识别的研究,到1986年汉字识别的研究进入一个实质性的阶段,不少研究单位相继推出了中文OCR产品.早期的OCR软件,由于识别率及产品化等多方面的因素,未能达到实际要求。同时,由于硬件设备成本高,运行速度慢,也没有达到实用的程度。只有个别部门,如信息部门、新闻出版单位等使用OCR软件。1986年以后我国的OCR研究有了很大进展,在汉字建模和识别方法上都有所创新,在系统研制和开发应用中都取得了丰硕的成果,不少单位相继推出了中文OCR产品。进入20世纪90年代以后,随着平台式扫描仪的广泛应用,以及我国信息自动化和办公自动化的普及,大大推动了OCR技术的进一步发展,使OCR的识别正确率、识别速度满足了广大用户的要求。

OCR 识别方式

与其它信息数据一样，在计算机中所有扫描仪捕捉到的图文信息都是用0、1。这两个数字来记录和进行识别的，所有信息都只是以0.1保存的一串串点或样本点。OCR识别程序识别页面上的字符信息主要通过单元模式匹配法和特征提取法两种方式进行字符识别。

单元模式匹配识别法(Pattern Matching)是将每一个字符与保存有标准字体和字号位图的文件进行不严格的比较。如果应用程序中有一个已保存字符的大数据库，则应用程序会选取合适的字符进行正确的匹配。软件必须使用一些处理技术找出最相似的匹配。通常是不断试验同一个字符的不同版本来比较。有些软件可以扫描一页文本，并鉴别出定义新字体的每一个字符。有些软件则使用自己的识别技术，尽其所能鉴别页面上的字符，然后将不可识别的字符进行人工选择或直接录入。

特征提取识别法(Feature Extraction)是将每个字符分解为很多个不同的字符特征，包括斜线、水平线和曲线等。然后又将这些特征与识别的字符进行匹配。特征提取法的优点是可以识别多种字体，例如中文书法体就是采用特征提取法实现字符识别的。

多数OCR应用软件都加入了语法智能检查功能，这种功能进一步提高了识别率。它主要通过上下文检查法实现拼写和语法的纠正。在文字识别时，OCR应用程序会做多次的上下文衔接性检查，根据程序中已经存在的词组、固定的用词顺序、对应的检查字符串的用词字。比较高级的应用软件会自动用它认为正确的词语替换错误词语，纠正语句意思.

主要识别流程：

影像倾斜校正：首先，OCR会检查原稿是否倾斜，如果原稿倾斜，此步驟將原稿旋转至水平或是垂直的位置，以提高辨识率。

图文分析：利用图文分析技术，判定文字区与图像区，区隔之后进一步撷取文字区的文字部分。

文字自动转正：选取部分文字区做0度、90度、180度、270度来选择辨识率最高的角度，然后依照辨识最高的角度，将整个原稿转正。

分割单字：将文章段落中的所有的文字、数码、标点符号与空格切割出来。

撷取单字的特征点做文字比对：OCR运用各种方法特征撷取技术来撷取出单字最特别、最明确的部分，尤其是可以用来辨明与其他单字不同的特征，进行与OCR文字资料库比对的动作，并察看这个编译过后的讯号，所相对应的最接近文字为何，而这个文字即是最后辨识的结果。

输出辨识结果：最后每一个文字都辨识完了之后，即产出这一次的OCR的文字档案。

TTS：

中文全称“从文本到语音”，它将计算机自己产生的、或外部输入的文字信息转变为可以听得懂的、流利的口语输出的技术，隶属于语音合成。语音合成为通过机械的、电子的方法产生人造语音的技术。TTS可以帮助有视觉障碍的人阅读计算机上的信息，或者只是简单的用来增加文本文档的可读性。

早在17世纪就有法国人研发机械式的说话设备。直到19世纪，贝尔实验室对于电子语音合成技术的研究，才打开近代语音合成技术的发展。贝尔实验室在1939年制作出第一个电子语音合成器VODER，是一种利用共振峰原理所制作的合成器。 1960年，瑞典语言学家G. Fant则提出利用线性预测编码技术（LPC）来作为语音合成分析技术，并推动了日后的发展。后来1980年代Moulines E和Charpentier F提出新的语音合成算法PSOLA，此技术可以合成比较自然的语音。

基本组成：

文本分析：对输入文本进行语言学分析（主要模拟人对自然语言的理解过程），逐句进行词汇的、语法的和语义的分析，以确定句子的低层结构和每个字的音素的组成，包括文本的断句、字词切分、多音字的处理、数字的处理、缩略语的处理等。使计算机对输入的文本能完全理解，并给出后两部分所需要的各种发音提示。

韵律建模：为合成语音规划出音段特征，如音高、音长和音强等，使合成语音能正确表达语意，听起来更加自然。

语音合成：根据韵律建模的结果，把处理好的文本所对应的单字或短语的语音基元从语音合成库中提取，利用特定的语音合成技术对语音基元进行韵律特性的调整和修改，最终合成出符合要求的语音。

中文 TTS系统

包括中文语音处理和语音合成，利用中文韵律等相关知识对中文语句进行分词、词性判断、注音、数字符号转换，语音合成通过查询中文语音库得到语音。目前中文TTS系统，比较著名的有：IBM，Microsoft，Fujitsu，科大讯飞，捷通华声等研究的系统。目前比较关键的就是中文韵律处理、符号数字、多音字、构词方面有较多的问题，需要不断研究，使得中文语音合成的自然化程度较高。

Node.js 介绍

Node.js是一个事件驱动I/O服务端JavaScript环境，基于Google的V8引擎。目的是为了提供撰写可扩充网络程序，如Web服务。第一个版本由Ryan Dahl于2009年发布。与一般JavaScript不同的地方，Node.js并不是在Web浏览器上运行，而是一种在服务器上运行的Javascript服务端JavaScript。

Javascript是一个事件驱动语言，Node利用了这个优点，编写出可扩展性高的服务器。Node采用了称为“事件循环(event loop)”的架构，使得编写可扩展性高的服务器变得既容易又安全。Node采用一系列“非阻塞”库来支持事件循环的方式。本质上就是为文件系统、数据库之类的资源提供接口。向文件系统发送一个请求时，无需等待硬盘（寻址并检索文件），硬盘准备好的时候非阻塞接口会通知Node。该模型以可扩展的方式简化了对慢资源的访问， 直观，易懂。

在Node启动的很短时间内，社区就已经贡献了大量的扩展库（模块）。其中很多是连接数据库或是其他软件的驱动，但还有很多是凭他们的实力制作出来的非常有用的软件。

Node.js 模块

Node.js使用Module模块去划分不同的功能，以简化应用的开发。Modules模块有点象C语言中的类库。每一个Node.js的类库都包含了十分丰富的各类函数，比如http模块就包含了和http功能相关的很多函数，可以帮助开发者很容易地对比如http,tcp/udp等进行操作，还可以很容易的创建http和tcp/udp的服务器。

要在程序中使用模块是十分方便的，只需要如下：

在这里，引入了http类库，并且对http类库的引用存放在http变量中了。这个时候，node.js会在我们应用中搜索是否存在node\_modules的目录，并且搜索这个目录中是否存在http的模块。如果node.js找不到这个目录，则会到全局模块缓存中去寻找，用户可以通过相对或者绝对路径，指定模块的位置，比如：

var myModule = require('./myModule.js');

模块中包含了很多功能代码片断，在模块中的代码大部分都是私有的，意思是在模块中定义的函数方法和变量，都只能在同一个模块中被调用。当然，可以将某些方法和变量暴露到模块外，这个时候可以使用exports对象去实现。

NPM 简介

NPM的全称是Node Package Manager，是一个NodeJS包管理和分发工具，已经成为了官方的发布Node模块的标准。

Nodejs自身提供了基本的模块，但是开发实际应用过程中仅仅依靠这些基本模块则还需要较多的工作。幸运的是，Nodejs库和框架为我们提供了帮助，让我们减少工作量。但是成百上千的库或者框架管理起来又很麻烦，有了NPM，可以很快的找到特定服务要使用的包，进行下载、安装以及管理已经安装的包。

本章小结

本章介绍了 Android 操作系统和 Cordova 跨平台开发框架，以及一些所涉及到的背景知识，为接下来的开发做好准备。

1. 报纸阅读器的需求分析

需求分析是指对要解决的问题进行详细分析，弄清楚问题的要求，包括输入、输出以及最终的结果。只有在明确了这个要求后，才能使最终开发出来的软件符合用户的需求。

业务描述

本课题的目标是开发一款基于 Android 系统的报纸阅读器，能够具备良好的兼容性、稳定性与实用性。所需要实现的基础功能如下：

拍照：调用手机自带的摄像头对报纸内容进行拍照。

裁剪内容：由于上一步得到的图片可能混入了一些与所需内容无关的部分，因此需要对图片进行裁剪。

OCR：将裁剪后的图片上传至远端的 OCR Cloud 引擎进行文字识别，并将识别结果返回给用户。

TTS：将识别所得的文字朗读出来。

存储：将 COR 是被后的文字存储在手机上，方便下一次使用，同时也需提供删除功能。

音量调节：可以对发音人的音量进行调大或者调小。

语音设置：进一步的个性化设置，可选择不同的地方性语言进行朗读。

业务流程处理

本课题主要为设计一款在 Android 系统上运行的报纸阅读器，并实现上述的所有功能。

用户打开该应用后，程序会自动检测手机中是否安装讯飞语音，该语音引擎在朗读文字的时候需要用到。如果检测到用户尚未安装，会弹出对话框提醒用户安装。安装完成后可以选择“拍照”或者“浏览”功能。

选择“拍照”后，程序调用系统自带的相机供用户拍摄报纸内容，待拍照完成后，跳转至图片裁剪界面。在该界面中会将用户在上一步拍摄的照片显示出来，并在图片中提供一个可任意方向及大小拖动的方框，用户可拖动该方框选出需要识别的图片部分。裁剪完成后点击“确定”即开始进行文字识别。由于该识别过程需要联网，如果发现用户为接入互联网会弹出对话框提醒用户进行相关设置。

在文字识别完成后，程序会自动跳转到朗读的界面。该界面会显示新闻的标题，并提供音量调节功能，拖动音量条即可对朗读的音量进行实时调节，并且程序会记录用户上一次所使用的音量大小，避免每次朗读时都要手动调大音量的麻烦。

改界面还提供语音设置与删除新闻的功能。语音设置功能由讯飞语音提供，可以更换不同的发音语言。

点击“浏览”跳转到历史新闻的界面，该界面以列表的形式显示了用户所有的识别记录。点击某条新闻即可跳转到朗读界面自动朗读。如果需要删除，长按该条新闻，会弹出对话框提醒用户是否真的需要删除，防止误操作。

用户需求分析

本课题的用户需求分析主要分为两个部分，一部分为核心需求，另一部分为潜在需求。

核心需求是支撑该应用的基本功能的需求。拍照、识别、朗读是本应用的核心需求。

潜在需求是指不容易发现但对于用户来说是很实用的需求。包括存储、音量调节、语音设置、删除前确认等。如果将识别后的文本存储起来，当用户下一次需要用到的时候就可以快速的浏览历史记录列表获得；由于手机在朗读时的音量可能偏小，老年人可能听不清，因此提供音量调节并记录上一次的音量大小的功能可以免却用户每次都要手动调整声音的麻烦；语音设置可以给予用户更多个性化的选择，如选择当地的方言进行朗读，更显亲切；删除前确认这项功能在大多数应用都有体现，主要是为了防止用户因某些原因所造成的误操作。

本章小结

本章主要描述了业务流程，分析了本课题的用户需求。业务流程列举了基本的业务功能，包括拍照、裁剪、OCR、TTS等。用户需求主要分析的各项功能的需求及存在的价值。

1. 报纸阅读器的功能设计

功能设计是按照产品定位的初步要求，在对用户需求及现有产品进行调查分析的基础上，对所定位产品应具备的目标功能系统进行概念性构建的创造活动。在上一章已经确定了用户需求，接下来需要在此基础上确定各个功能模块的定义划分。

客户端业务处理流程

在上一章的讨论中，本课题的主要功能是拍照、裁剪、OCR、TTS。

UI 组件布局

HTML、CSS 简介

HTML 中文全称“超文本标记语言”，“超文本”指页面内可以包含图片、链接、音乐和视频等非文字元素。其结构分为 “head”与 “body”两部分，其中“头”部提供关于网页的信息，“主题”部分提供网页的具体内容。HTML 具有简易性、可拓展性、平台无关性和通用性的特点。一个 HTML 文件以 .htm 或者 .html 为拓展名。在2014年10月29日，HTML5 标准规范制定完成。其加入了很多新特性：语义特性、本地存储特性、设备兼容特性、性能与集成特性、连接特性、3D图形特效等。

CSS 中文全称“层叠样式表”，是用来表现 HTML 等文件样式的计算机语言。相对于传统 HTML 的表现而言，CSS 能够对网页中的对象的位置排版进行像素级的精确控制，支持几乎所有的字体字号样式，拥有对网页对象和模型样式编辑的能力，并能够进行初步交互设计，是目前基于文本展示最优秀的表现设计语言。CSS 能够根据不同使用者的理解能力，简化或者优化写法，针对开发者而言有较强的易读性。 CSS3是CSS技术的升级版本，CSS3语言开发是朝着模块化发展的。以前的规范作为一个模块实在是太庞大而且比较复杂，所以，把它分解为一些小的模块，更多新的模块也被加入进来。包括盒子模型、列表模块、超链接方式 、语言模块 、背景和边框 、文字特效 、多栏布局等。

一个使用 HTML 与 CSS 的简单的布局文件可以是这样的：

< !DOCTYPE html>

<head>

<style type="text/css">

body {background-color: yellow}

h1 {background-color: #00ff00}

h2 {background-color: transparent}

p {background-color: rgb(250,0,255)}

p.no2 {background-color: gray; padding: 20px;}

</style>

</head>

<body>

<h1>这是标题 1</h1>

<h2>这是标题 2</h2>

<p>这是段落</p>

<p class="no2">这个段落设置了内边距。</p>

</body>

</html>

每个 HTML 文件都以文件类型声明开头，里层包括 <head></head> 和 <body></body>标签。<body></body> 包含了整个文档的可见部分。

UI 布局

总体设计风格：

本软件使用类似 iOS7 风格的扁平化极简化设计，提高了辨识度与易用性。这种风格最主要的特点是弱化操作元素，突出内容，使其清晰可见。遵循 UI服务于内容而不影响内容的表现，因此弱化了控件的视觉效果，以避免干扰内容；同时去除了控件的纹理和质感的表现，非必要时不再使用拟物化的方式来表达，取而代之的是简化的图标元素和操作内容。这种设计理念有助于营造纵深感，建立层次结构和秩序，并帮助用户理解屏幕元素间的物理关系。

主界面：

主界面放置两个由上而下层叠的按钮，每个按钮的高度为整个窗口高度的 50%， 宽度占满整个屏幕的宽度，用辨识度较高的两种颜色作为背景色以便于区分，按钮的提示文字均在水平和垂直方向上居中，分别提供的功能是拍照和浏览。布局下所示：

HTML：

<div id="takePhoto"><span>拍 照</span></div>

<div id="browserNews"><span>浏 览</span></div>

CSS：

#takePhoto, #browserNews {

display: table;

width: 100%;

height: 50%;

text-align: center;

font-weight: bold;

font-size: 50px;

color: white;

}

#takePhoto span,

#browserNews span {

display: table-cell;

vertical-align: middle;

}

#takePhoto {

background-color: #26a69a;

}

#browserNews {

background-color: #ee6e73;

}

拍照界面：

该部分使用系统提供的 UI，暂不进行自定义设置。

裁剪界面：

该部分会显示一张图片，图片中有可随意拖动的选框用于裁剪。图片下方放置两个按钮，供用户确定或取消裁剪操作。

HTML：

<div id="container">

<img id="crop" src="" alt="" style="display: none;"/>

</div>

<div id="control">

<button id="cancel">取消</button>

<button id="submit">确定</button>

</div>

朗读界面：

该界面会显示当前所朗读的新闻的标题，下面放置了一系列按钮用于音量调节、重复朗读、语音设置和删除新闻。

HTML：

<h4 id="speaking-title"></h4>

<div id="tip-container">

<button id="delete" class="btn btn-block btn-negative">删除</button> </div>

<div id="volume-container">

<button class="btn btn-block btn-outlined">音量调节</button>

<div id="range-container">

<input type="text" name="range" id="range" style="display: none;"/> </div>

</div>

<div id="tts-container">

<button id="tts" class="btn btn-block btn-outlined">语音设置</button> </div>

<div id="repeat-container">

<button id="repeat" class="btn btn-block btn-outlined">重新朗读</button>

</div>

新闻列表界面：

该界面主要由一竖向列表组成，每个列表项均为可点击的链接，点击后回跳转至朗读界面。

HTML：

<ul id="news" class="table-view" style="display: none;">

<li class="table-view-cell" data-id="%id%"><a class="navigate-right" href="%link%">%title%</a></li>

<li class="table-view-cell" data-id="%id%"><a class="navigate-right" href="%link%">%title%</a></li>

<li class="table-view-cell" data-id="%id%"><a class="navigate-right" href="%link%">%title%</a></li>

</ul>

拍照功能的调用：

使用 Cordova 插件 org.apache.cordova.camera 即可用 JavaScript 的方式调用到系统的摄像头。其提供了 navigator.camera.getPicture(cameraSuccess, cameraError, cameraOptions ); 这个 API 来使用，允许开发者对拍照时的多项参数进行设置，包括图片质量、图片种类、宽高大小等，这些配置项在 cameraOptions 中可以调整。

裁剪图片：

使用 jQuery 插件 cropper 实现。其具有多项特性因而可以在裁剪图片时提供较好的操作体验，这些特性包括支持移动设备上的触摸事件、支持旋转以及缩放图片等，而且其库文件体积较小，更适合在移动设备上使用。

上传图片：

使用 Cordova 插件 org.apache.cordova.file-transfer 插件实现。在设定好上传地址以及 POST 参数后，即可使用 FileTransfer 提供的实例方法 upload 进行图片上传，该方法还接受两个回调函数作为参数，用于上传成功和失败的处理。

当上传成功且服务器端完成了 OCR 识别后，会将识别结果返回，此时便会触发上传成功的回调函数，该函数具有一个包含响应结果的对象作为参数，该对象的

response 方法即为从服务器返回的识别结果。

OCR 识别：

使用 ABBYY 提供的 ABBYY Cloud OCR SDK 实现。ABBYY 是一家俄罗斯软件公司，在文档识别，数据捕获和语言技术的开发中居世界领先地位。它不仅支持多国文字，还支持彩色文件识别、自动保留原稿插图和排版格式以及后台批处理识别功能等，而且还能将识别结果保存成RTF、TXT、DOC、CSV、XLS或HTML等格式，并且它能保持表格与图片中原始的多栏页面设计。

ABBYY Cloud OCR SDK 提供了多种方式给开发者使用其 Cloud OCR 引擎，包括 .NET、cURL、JAVA、PHP、Python、ASP.NET、Ruby、JavaScript等，因此也可以在 Android、iPhone 和 Windows Phone 等平台中使用。官方文档中均有提供相应开发语言的代码例程，开发者可以很方便的使用和修改。

TTS：

使用 Cordova 插件 com.wordsbaking.cordova.tts 实现。Android 系统从 4.0.3 版本（API Level 15）及以后开始支持 TTS 服务，因此该插件必须在 Android 4.0.3 及之后的版本才可使用。该插件支持多项朗读设置，包括朗读语言、语速等。

本章小结

本章主要分析了各项功能模块的业务处理流程，以及在实现各模块时所用的相应插件与服务，和使用时的参数功能调整。这些模块包括 UI 界面、拍照、裁剪、上传、OCR、TTS等。

1. 报纸阅读器的实现

本章将详细讲解各项功能模块的具体代码实现以及相应的运行情况。

项目开发软硬件平台

硬件平台：

MacBook Air 笔记本一台

处理器：1.4 GHz Intel Core i5

内存：8 GB 1600 MHz DDR3

图形卡：Intel HD Graphics 5000 1024 MB

Android 智能手机一部

型号：红米手机

处理器：联发科 MT6589T

内存：8 GB

运行内存：1 GB

操作系统：Android 4.2.1

USB数据线一根

SIM 卡一张

软件平台：

操作系统：OS X 10.9.5

服务器平台：LeanCloud

程序语言：HTML、CSS、JavaScript

开发包：JAVA u845， Android SDK 4.4.2(API 19)

开发调试工具：WebStorm、npm、Android Development Tools、Chrome 浏览器

项目创建：

开发环境搭建：

本项目需要安装 Android 开发环境以及 Cordova 框架。下面将分为 JDK、Android SDK、Node.js、Cordova 4个部分进行讲解。

安装 JDK

编译 Android 时需要用到 JRE 支持。安装 JDK 需要在 Sun 官方网站进行下载，地址是 <http://www.oracle.com/technetwork/java/javase/downloads/index.html> ，这里选择 Java SE 8u45 下载。然后双击下载后的文件即可自动安装。安装完成后打开终端，并输入 java –version ，即可看到相应的 Java 安装信息。默认的安装目录实在 /System/Libray/Frameworks /JavaVM.Framwork/，在该目录下 Versions 文件夹里存放了不同版本的 jdk。

安装 Android SDK

前往 Android 开发者网站下载，地址是 developer.android.com 。下载后解压即可使用。

安装 Node.js

前往 Node.js 官方网站进行下载，地址是 <https://nodejs.org/> 。这里选择的是 Node.js for Mac的安装文件进行下载，下载完成后双击下载文件即可自动开始安装。安装完成后在终端中输入 node –v 即可显示当前 Node.js 的安装版本号。

安装 Cordova

在电脑打开终端，输入 sudo npm install -g cordova 开始安装 Cordova。安装完成后在终端输入 cordova –v 即可显示当前 cordova 的版本号。

Cordova 工程创建

1. 打开终端，输入 cordova create SmartReader com.xyq.smartreader SmartReader，然后按回车即可成功创建 Cordova 项目。第一个参数是指创建该工程的文件夹名，如果不存在将会创建该文件夹。第二个参数为该项目的包名。第三个参数为该项目在手机下显示的名字。
2. 在终端输入 cd SmartReader，进入该项目的文件夹。

输入 cordova platform add android ，添加 Android 平台支持。Cordova 会调用 Android SDK在platforms/android中创建一个完整的 Android 应用工程。

1. 在 Webstorm 中打开 SmartReader 文件夹，即可看到该目录下已经生成了一系列的文件。

完成创建 Cordova 工程。Cordova会在 www 目录默认生成index.html, css/index.css, img/logo.png 和 js/index.js 等文件。

6.将手机通过 USB 数据线与笔记本电脑连接，并开启开发者选项以及使用 USB 调试的功能。在终端中输入 cordova run android 即可将该项目运行在手机上。如果需要在模拟器中运行，则在终端输入 cordova emulate android 即可启动 Android 模拟器。

项目目录介绍：

在使用Cordova创建的项目目录下，platforms文件夹中存放了Cordova生成的相应平台的应用工程。例如使用命令行cordova platform add android添加了对Android平台的支持后，platforms文件夹会自动生成android目录，里面存放的是Cordova生成的Android项目。plugins目录下存放了开发者添加的插件。www目录里存放的是HTML、CSS、JavaScript以及一些显示用的图片。

功能实现

UI

由于 Cordova只提供了调用Native API的接口，并没有提供任何UI方面的功能，所以UI方面这里使用 Ratchet 框架来构造界面。Ratchet 是一款专门为移动开发打造的框架，包括按钮、列表、标签栏等在移动设备上常见的组件，让开发者可以很方便的制作精美的页面。其支持 Android 与iOS7两种风格，本软件使用iOS7风格。首先需要到 Ratchet 官方网站上下载该框架，下载完成后解压，将相应的 JavaScript 和 CSS 文件复制到项目目录下 www 文件夹的对应目录。然后即可引用：

<link rel="stylesheet" href="ratchet/css/ratchet.min.css"/>

拍照功能

由于从 Cordova 3开始采用插件的方式提供核心 API，因此需要通过安装插件的方式进行拓展。首先安装 Cordova 插件 org.apache.cordova.camera，该插件提供对系统照相机功能调用的接口。在终端输入 cordova plugin add org.apache.cordova.camera ，安装该插件。该插件提供了 navigator.camera.getPicture 这个方法来调用系统的拍照功能，该方法需要传入三个参数，分别是：拍照成功后的回调函数、拍照失败后的回调函数和拍照的参数。拍照的参数包括图片质量（默认为75%）、图片宽高、图片拓展名等。具体使用代码如下：

navigator.camera.getPicture(onSuccess, onFail, {

quality: 100,

destinationType: Camera.DestinationType.FILE\_URI

});

function onSuccess(imageURI) {

localStorage.setItem("imgUrl", imageURI);

location.href = "crop.html";

}

function onFail(error) {}

在拍照完成后需要跳转到裁剪图片的页面，因此在跳转前需要把图片的地址保存下来以供下一步使用。

裁剪图片：

使用 Image Cropper 插件实现。首先需要到该插件官网 <https://github.com/fengyuanchen/cropper> 下载。下载完成后将文件解压，将相应的 JavaScript 文件 和 CSS 文件放到 SmartReader 项目下的 www 文件夹中对应的位置。由于该插件依赖 jQuery，因此在引入其 JavaScript 前需要先引入 jQuery。引入代码如下：

<link rel="stylesheet" type="text/css" href="css/crop.css" />

以及：

<script src="js/jquery.min.js"></script>

<script src="js/cropper.js"></script>

引入完成后即可开始使用。该插件提供了一个名为cropper 的方法供开发者调用裁剪功能，该方法需要传入一个为 Object形式的配置项参数。可选的配置项包括图片比例、默认裁剪框的百分比、图片缩放、限制裁剪大小等。

首先需要在 HTML 文件中放置一个 img 标签供该插件使用：

<div id="container">

<img id="crop" src="" alt="" style="display: none;"/>

</div>

裁剪图片：

$("#crop").cropper({

autoCropArea: 0.85

});

获得裁剪后的图片信息，为 DataURL 形式：

var dataURL = $("#container #crop").cropper("getCroppedCanvas").toDataURL();

上传图片：

需要使用 Cordova 插件 org.apache.cordova.file-transfer。在终端中输入 cordova plugin add org.apache.cordova.file-transfer，即可安装。该插件提供了全局的FileTransfer和FileUploadOptions构造函数，使用时需要先实例化。实例化FileUploadOptions后需要配置fileKey，这是一个字符串，需要跟服务器端统一，这样服务器端才可根据该 Key值取到相应的上传图片。实例化FileTransfer后需要传入五个参数，分别是上传的数据、服务器地址、上传成功的回调、上传失败的回调和FileUploadOptions实例化对象。上传代码如下：

var options = new FileUploadOptions();

options.fileKey = "ocrImage";

options.chunkedMode = false;

var uri = encodeURI("http://dev.paper-reader.avosapps.com/upload");

var ft = new FileTransfer();

ft.upload(dataURL, uri, function (r) {

window.plugins.spinnerDialog.hide();

var text = r.response;

localStorage.setItem("text", text);

location.href = "speak.html";

}, function (error) {

//todo

//alert("error " + error);

}, options);

在获取 OCR 识别后的文字，需要该文件存储在 localStorage 中，以便下一次的使用。这一步的操作将放到跳转到朗读页面后进行。存储的内容包括新闻标题、正文以及一个全局 id，全局id 用于在朗读或者删除时在 localStorage 中找到对应的内容。默认将文本的第一行作为标题，其余部分为正文。

var obj = {};

var current = parseInt(localStorage.getItem("current")) + 1;

title = text.split("\n")[0];

var content = text.split("\n").slice(1).join("");

id = current;

obj.id = current;

obj.title = title;

obj.content = content;

newsList = JSON.parse(localStorage.getItem("news"));

newsList.push(obj);

localStorage.setItem("news", JSON.stringify(newsList));

localStorage.setItem("current", current);

浏览新闻：

需要先将所有存储在 localStorage 中的新闻条目取出，再逐条渲染在一个列表中。该页面提供两种操作：点击新闻，即可跳转到朗读界面，自动朗读该条新闻；

长按新闻，将会弹出一个对话框询问是否删除该条新闻，如果选择“确定”将会把该条新闻从列表和 localStorage 存储中移除。

var newsList = JSON.parse(localStorage.getItem("news"));

var newsElem = document.getElementById("news");

var template = '<li class="table-view-cell" data-id="%id%"><a class="navigate-right" href="%link%">%title%</a></li>';

var elemList = [];

if (newsList.length == 0) {

document.getElementById("tip-container").style.display = "block";

return;

}

for (var i = 0, length = newsList.length; i < length; i++) {

var id = newsList[i].id;

var title = newsList[i].title;

var elemStr = template.replace("%link%", "speak.html?id=" + id)

.replace("%title%", title)

.replace("%id%", id);

elemList.push(elemStr);

}

newsElem.innerHTML = elemList.join("");

newsElem.style.display = "block";

TTS：

需要使用 Cordova 插件 com.wordsbaking.cordova.tts。在终端中输入 cordova plugin add com.wordsbaking.cordova.tts，即可安装。该插件提供TTS.speak这个方法用于朗读指定的文本。使用该方法需要传入三个参数，分别是朗读配置项、朗读成功后的回调函数和朗读失败后的回调函数。其中朗读配置项是一个Object形式的参数，可设置朗读文本、朗读语音和朗读速率。如果需要停止朗读，则需调用TTS.stop()这个方法。

TTS.speak({

text: text,

locale: "zh-CN",

rate: 0.75

}, function () {

}, function () {

});

音量调节：

需要使用 Cordova 插件 com.develcode.plugins.volumeControl。在终端中输入 cordova plugin add git+https://github.com/manusimpson/Phonegap-Android-VolumeControl.git，即可安装。该插件提供了名为VolumeControl.setVolume的方法用于控制音量。

使用该方法是需要传入三个参数，分别是目标音量大小、调整音量成功后的回调函数和调整音量失败后的回调函数。如果需要获取当前音量大小，则使用 VolumeControl.getVolume 方法即可。

VolumeControl.setVolume(parseInt(elem.value), onVolSuccess, onVolError);

function onVolSuccess(){

//alert("Volume changed");

}

function onVolError(){

//Manage Error

}

检测网络状态：

需要使用 Cordova 插件 org.apache.cordova.network-information。在终端中输入 cordova plugin add org.apache.cordova.network-information，即可安装。通过获取 navigator.connection.type 该属性的值即可得知用户实时的网络状况。由于这里仅需要在用户无网络连接时给出提醒，因此仅需判断 navigator.connection.type 是否 为 “none” 即可。代码如下：

var networkState = navigator.connection.type;

if (networkState == "none") {

navigator.notification.confirm(

"检测到您已经断开网络，是否马上打开？",

onConfirm,

"未联网",

["取消", "去打开"]

);

function onConfirm(buttonIndex) {

if (buttonIndex == 2) {

LocationAndSettings.switchToWifiSettings();

}

}

显示对话框：

需要使用 Cordova 插件 org.apache.cordova.dialogs。在终端中输入 cordova plugin add org.apache.cordova.dialogs，即可安装。该插件提供了 alert、confirm、prompt和beep四种提醒方式。这里仅需用到 prompt 方式。对应的方法名为navigator.notification.confirm。使用该方法需要传入四个参数，分别是对话框提示内容、回调函数、对话框提示标题和按钮文本。其中在回调函数中可以获取用户点击了哪个按钮从而触发相对应的操作。使用代码如下：

navigator.notification.confirm(

"检测到您已经断开网络，是否马上打开？",

callback,

"未联网",

["取消", "去打开"]

);

音量条：

需要使用插件 Powerange。这是一款具有 iOS7 风格的滑块插件。首先需要到该插件官网<http://abpetkov.github.io/powerange/> 下载。完成后解压，并将相应的 JavaScript文件和CSS文件拷贝到项目文件夹www目录中对应的文件夹。该插件提供了名为Powerange的构造函数。实例化时需要传入两个参数，分别是用于绑定的HTML Input元素和配置项。其中配置项包括一系列可选的参数，包括移动滑块后触发的回调函数、滑块的取值范围、起始位置等。使用代码如下：

<div id="range-container">

<input type="text" name="range" id="range" style="display: none;"/>

</div>

var elem = document.getElementById("range");

new Powerange(elem, {

min: 0,

max: 100,

start: parseInt(localStorage.getItem("volume")),

callback: function() {

VolumeControl.setVolume(parseInt(elem.value), onVolSuccess, onVolError);

localStorage.setItem("volume", elem.value + "");

}

});

检测用户是否安装讯飞语音：

需要使用 Cordova 插件 com.ohh2ahh.plugins.appavailability。在终端输入 cordova plugin add com.ohh2ahh.plugins.appavailability，即可安装。该插件提供了名为appAvailability.check的方法用于检测特定的APP是否有被安装在手机上。使用该方法时需要传入三个参数，分别是APP的包名、检测到成功安装的回调函数和检测到未成功安装的回调函数。使用代码如下：

appAvailability.check(

' com.iflytek.tts ', // Package Name

function() { // Success callback

},

function() { // Error callback

}

);

获取默认语音引擎名称：

需要使用 Cordova 插件 com.wordsbaking.cordova.tts。在前面已经安装该插件。这里直接使用其检测方法TTS.getDefaultEngine。使用该方法需要传入两个参数，分别是成功获取时的回调函数和出错的回调函数。在成功获取时的回调函数中可以取得默认的TTS引擎的包名。使用代码如下：

TTS.getDefaultEngine(function (name) {

if (name == "com.iflytek.speechcloud" || name == "com.iflytek.tts") {

} else {

}

}, function (error) {

//alert(error)

});

跳转至设置页面：

需要使用 Cordova 插件 sk.tamex.locationandsettings。在终端中输入 cordova plugin add sk.tamex.locationandsettings，即可安装。该插件提供了一系列的设置功能，包括对Wifi、GPS和TTS等。这里仅需要使用跳转至Wifi和TTS设置界面的功能即可。

跳转至 Wifi 设置界面，使用如下方法：

LocationAndSettings.switchToWifiSettings();

跳转至 TTS设置界面，使用如下方法：

LocationAndSettings.switchToTTSSettings();

数据存储：

HTML5提供了两种在客户端存储数据的新方法，分别是localStorage和sessionStorage。在这之前，这类功能都是使用 cookie实现的。但是cookie并不适合大量数据的存储，一般情况下只能存储大约4KB的数据。localStorage对数据的存储没有时间限制，sessionStorage存储的数据在用户关闭浏览器窗口后会给删除。因而在本应用中实现存储功能应使用localStorage。localStorage使用键值对的方式进行数据存储，目前仅支持存储字符串类型的数据。其提供了 setItem、getItem和removeItem三个方法，分别对应的是写、读、删的功能。

兼容性测试：

兼容性（compatibility）是指硬件之间、软件之间或是软硬件组合系统之间的相互协调工作的程度。兼容的概念比较广，对于软件的兼容性，指的是指某个软件能稳定地工作在若干个操作系统之中，而不会出现意外的退出等问题。

这里使用阿里移动测试服务对本应用进行基本的兼容性测试，地址是http://mts.aliyun.com。阿里移动测试服务是为广大移动开发者提供真机测试服务的云平台，拥有大量热门机型，提供7x24全天候服务。能够帮助开发者以最低的成本、最高的效率发现APP中的各类隐患，包括APP崩溃、各类兼容性、功能性、性能问题等。

测试使用的手机设备包括三星、华为、小米和魅族一共30部手机，测试的Android操作系统版本从4.0.4 到4.4.4，已经涵盖了市面上大多数的Android热门机型，因此其测试结果也是具有一定的实际意义的。

模块功能测试：

功能测试，也称为行为测试，根据产品特性、操作描述和用户方案，测试一个产品的特性和可操作行为以确定它们满足设计需求。功能测试是为了确保程序以期望的方式运行而按功能要求对软件进行的测试，通过对一个系统的所有的特性和功能都进行测试确保符合需求和规范。功能测试也叫黑盒测试或数据驱动测试，只需考虑需要测试的各个功能，不需要考虑整个软件的内部结构及代码.一般从软件产品的界面、架构出发，按照需求编写出来的测试用例，输入数据在预期结果和实际结果之间进行评测，进而提出更加使产品达到用户使用的要求。

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 用例名称 | 测试目的 | 测试方法 |
| 1 | APP首页 | 测试该APP能否正常打开 | 打开该APP，观察能否进入到首页 |
| 2 | 拍照 | 测试能否调用系统摄像头进行拍照 | 点击“拍照”后观察能否进入系统拍照界面 |
| 3 | 裁剪图片 | 测试能否正常裁剪图片 | 拍照完成后进入裁剪界面，使用选框对图片进行任意尺寸的裁剪 |
| 4 | 上传图片 | 测试能否正常上传图片 | 裁剪完成后选择“确定”进行上传图片，观察能否从服务器返回结果 |
| 5 | 上传前断网提醒 | 测试在上传前断网能否给与用户提醒并提供跳转到网络设置的功能 | 上传前将手机网络断开，观察是否会有对话框提醒已断网，以及是否提供跳转至网络设置界面的功能 |
| 6 | 识别完成后自动朗读 | 测试在识别后可否自动朗读结果 | 上传图片待返回结果后观察能否在跳转至朗读页面后自动朗读 |
| 7 | 音量调节 | 测试可否在调整朗读时的音量大小 | 在朗读时使用音量条调整音量，观察音量大小是否发生变化 |
| 8 | 自动记住上一次调整的音量大小 | 测试可否自动记住上一次调整后的音量大小 | 在朗读时调整音量大小，然后重新进入朗读界面，观察音量大小是否跟上一次一致 |
| 9 | 重新朗读 | 测试可否重新朗读 | 在朗读界面点击“重新朗读”按钮，观察是否从头开始朗读 |
| 10 | 语音设置 | 测试可否跳转至语音设置界面 | 在朗读界面点击“语音设置”按钮，观察是否跳转至系统语音设置界面 |
| 11 | 在朗读界面删除当前新闻 | 测试可否在朗读界面删除当前新闻 | 在朗读界面点击“删除”按钮确认删除后，再进入浏览新闻的界面，观察该条新闻可否被找到 |
| 12 | 在朗读界面删除当前新闻前提醒 | 测试可否在朗读界面删除当前新闻前提醒用户确认该操作 | 在朗读界面点击“删除”按钮后，观察是否会有对话框提醒用户是否确认删除 |
| 13 | 讯飞语音检测检测 | 测试可否检测用户是否安装了讯飞语音 | 在手机上未安装讯飞语音的情况下打开该APP，观察是否有对话框弹出提醒用户安装讯飞语音 |
| 14 | TTS默认引擎检测 | 测试可否检测用户是否将讯飞语音设置为默认TTS引擎 | 在安装讯飞语音但未将其设置为默认TTS引擎的情况下打开该APP，观察是否有对话框弹出提醒用户去设置，并提供跳转到设置界面的功能 |
| 15 | 浏览新闻列表 | 测试可否使用浏览新闻功能 | 在首页点击“浏览”后观察可否跳转到新闻列表界面 |
| 16 | 朗读新闻列表内的新闻 | 测试可否正常朗读新闻列表内的新闻 | 在新闻列表点击新闻，观察是否会跳转到朗读界面并自动将该条新闻朗读出来 |
| 17 | 在新闻列表删除新闻 | 测试可否在新闻列表删除新闻 | 在新闻列表长按新闻条目，观察是否会有对话框弹出提醒确认删除，以及删除后再次进入新闻列表页面，观察该条新闻可否被找到 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

OCR识别测试：

应用效果：

本章小结

本章主要详细讲解了各项功能模块的具体实现，包括插件安装以及实现代码，以及一系列的测试。