**深 圳 大 学**

**联合培养**

**深圳职业技术学院**

本 科 毕 业 论 文（设计）

题目: **移动web电子阅读器**

姓名: **李俊瑾**

专业: **计算机科学与技术**

学院: **计算机工程学院**

学号: **13190023**

指导教师: **秦高德**

职称： **副教授**

2017 年 4 月 6 日

**深圳大学深圳职业技术学院联合培养**

**本科毕业论文（设计）诚信声明**

本人郑重声明：所呈交的毕业论文（设计），题目《**移动web电子阅读器**》 是本人在指导教师的指导下，独立进行研究工作所取得的成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式注明。除此之外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。本人完全意识到本声明的法律结果。

毕业论文（设计）作者签名：

日期： 年 月 日

目录

[1.绪论 1](#_Toc479245394)

[1.1 项目背景 1](#_Toc479245395)

[1.2 国内现状 1](#_Toc479245396)

[1.3 国外现状 2](#_Toc479245397)

[2.概述 2](#_Toc479245398)

[2.1 课题概述 2](#_Toc479245399)

[2.2 相关技术 2](#_Toc479245400)

[2.2.1 HTML5 2](#_Toc479245401)

[2.2.2 CSS3 2](#_Toc479245402)

[2.2.3 Zepto.js 3](#_Toc479245403)

[2.2.4 Vue.js 3](#_Toc479245404)

[2.2.5 Node.js 3](#_Toc479245405)

[2.2.6 Koa1.0 3](#_Toc479245406)

[2.2.7 MongoDB 4](#_Toc479245407)

[2.3 论文组织结构简介 5](#_Toc479245408)

[2.3.1 绪论 5](#_Toc479245409)

[2.3.2 概述 5](#_Toc479245410)

[2.3.3 系统分析 5](#_Toc479245411)

[2.3.4 系统设计 5](#_Toc479245412)

[2.3.5 系统实现方案 5](#_Toc479245413)

[2.3.6 系统核心功能实现 5](#_Toc479245414)

[2.3.7 系统测试 6](#_Toc479245415)

[2.3.8 结束语 6](#_Toc479245416)

[3.系统分析 6](#_Toc479245417)

[3.1 可行性分析 6](#_Toc479245418)

[3.1.1 技术可行性 6](#_Toc479245419)

[3.1.2 经济可行性 6](#_Toc479245420)

[3.1.3社会可行性 6](#_Toc479245421)

[3.2 开发环境 7](#_Toc479245422)

[3.2.1 硬件平台 7](#_Toc479245423)

[3.2.2 软件平台 7](#_Toc479245424)

[3.3 需求分析 9](#_Toc479245425)

[3.3.1 阅读器开发需求 10](#_Toc479245426)

[3.3.2 书城展示列表需求 12](#_Toc479245427)

[3.3.3 管理员后台开发 13](#_Toc479245428)

[3.3.4 服务端接口开发 13](#_Toc479245429)

[3.3.5 系统性能需求 14](#_Toc479245430)

[3.4 系统设计 14](#_Toc479245431)

[3.4.1 总体设计目标 14](#_Toc479245432)

[3.4.2 系统功能描述 15](#_Toc479245433)

[3.4.3 系统功能模块图 15](#_Toc479245434)

[3.4.4 业务流程图 18](#_Toc479245435)

[3.4.5 数据字典 19](#_Toc479245436)

[3.4.6 数据流图 22](#_Toc479245437)

[4. 系统实现步骤 23](#_Toc479245438)

[4.1 开发环境准备 23](#_Toc479245439)

[4.2 开始项目 23](#_Toc479245440)

[4.2.1项目文件搭建 23](#_Toc479245441)

[4.2.2项目依赖初始化 24](#_Toc479245442)

[4.2.3项目入口文件app.js 25](#_Toc479245443)

[4.2.4 创建视图层文件 26](#_Toc479245444)

[4.2.5 页面路由开发及测试前端流程 27](#_Toc479245445)

[4.2.6 设计数据库模型 28](#_Toc479245446)

[4.2.7 在MongoDB客户端查看数据 29](#_Toc479245447)

[4.2.8 后端业务逻辑开发 29](#_Toc479245448)

[4.2.9 页面自适应屏幕宽度代码改造 30](#_Toc479245449)

[4.2.10 项目托管 31](#_Toc479245450)

[5. 系统核心功能实现方法 33](#_Toc479245451)

[5.1 首页轮播图实现 33](#_Toc479245452)

[5.2 字体大小及背景颜色记忆功能 35](#_Toc479245453)

[5.3 用户注册功能的实现 36](#_Toc479245454)

[5.4 后端数据渲染至前端的实现 39](#_Toc479245455)

[5.5 阅读器翻页功能实现 43](#_Toc479245456)

[6. 系统测试 44](#_Toc479245457)

[6.1 测试目的 44](#_Toc479245458)

[6.2 测试对象 44](#_Toc479245459)

[6.3 测试流程 44](#_Toc479245460)

[6.3.1测试注册功能 44](#_Toc479245461)

[6.3.2测试管理员添加功能 46](#_Toc479245462)

[6.3.3测试搜索功能 47](#_Toc479245463)

[6.3.4测试阅读器字体操作面板功能 48](#_Toc479245464)

[6.4 测试结论 49](#_Toc479245465)

[7. 结束语 50](#_Toc479245466)

[【参考文献】 51](#_Toc479245467)

[致谢 52](#_Toc479245468)

**移动web电子阅读器**

计算机工程学院 计算机科学与技术

李俊瑾 学号：13190023

**【摘要】**在这个移动互联网时代，移动设备在我们生活中扮演者越来越重要的角色，也越来越多的系统会开发成Web App[1]的形式，因为其轻量便捷特性受到大众的喜爱。本文论述的是一个移动web电子阅读器系统，本系统是一个基于HTML5**[2]**,CSS3**[3]**,Zepto.js**[4]**来开发的Web App,开发过程分为四大部分，前端阅读器界面开发，前端书城界面开发，管理员界面开发以及服务端接口开发。用户可以注册登录，阅读书籍，管理员可以添加书籍详情，删除书籍。本系统服务端使用Node.js**[5]**框架Koa**[6]**进行快速搭建。本系统是针对使用手机阅读书籍的用户开发的，方便用户随时随地使用手机阅读。

**【关键字】**web app；电子阅读器；HTML5;CSS3;Zepto.js；Koa；MongoDB

# 1.绪论

## 1.1 项目背景

在当今时代，随着社会的迅猛发展，互联网的趋势渐渐从PC时代转向移动互联时代，越来越多人的衣食住行都依赖于移动设备。“一机在手，行走天下”的生活方式渐渐在影响着大众，手机以其自身携带方便，功能齐全的优势，成为人们生活中必不可少的工具。因此基于移动端的产品开发成为一种潮流。

在这个大众的衣食住行基本上都得到了满足的年代，越来越多的人开始转向精神方面的享受。比如阅读，旅行，音乐，电影等精神层面的艺术熏陶。生活节奏越来越快，为了保证用户适应快节奏生活的同时又能让用户享受到精神粮食，基于这方面的开发研究显然是一个不错的选择。

本次选题是移动web电子阅读器，是一款运行在用户手机浏览器的Web App，用户只要点击操作手机，即可进行阅读小说及其它相关操作，Web App因其跨平台，无需安装，更新简单而受到用户欢迎。

## 1.2 国内现状

阅读对于国民来说是一件重要的事情。据新闻报导，2015年我国成年国民图书阅读率**[7]**为58.4%，数字媒介阅读率为64.0%，在数字媒介中有60.0%的人是通过手机进行阅读的，也就是说在通过数字媒介阅读的国民中有超过一半的人是通过手机来阅读的，在我们日常生活中，出门坐地铁，使用手机阅读，打发时间也成为了一种流行。

当今国内也有很多知名的电子阅读器APP，如掌阅科技有限公司的掌阅APP，腾讯旗下的QQ阅读。

## 1.3 国外现状

国外也越来越多的人阅读方式在改变，渐渐从纸质书走向电子书，流行于阅读手机小说，据调查发现在欧美阅读电子书的用户中，在阅读终端的选择中，更偏向于选用手机来阅读，因为比起平板，电脑，阅读器来说，手机功能更多，携带更方便。

# 2.概述

## 2.1 课题概述

基于上述的背景以及国内外现状，开发一个适合当今社会流行趋势，符合大众需求的阅读APP是一件必要的事情。本系统是一个轻量级的电子阅读器Web App，Web App是区别于原生Native App及混合Hybrid App的一种运行在浏览器端的网页，它优势是轻量级，不需要像原生APP那样经过应用商店审核才更新，可以直接更新版本。本次开发的阅读器分别有两个面向用户的主要界面：书籍列表展示界面和阅读器，可以让用户方便快捷地在浏览器端打开，阅读小说。

## 2.2 相关技术

本系统移动web电子阅读器按照功能模块开发，分别为阅读器界面，前台书籍展示界面，管理员编辑界面和后台接口开发，前端数据从后台接口获取。

系统中前端框架部分是基于HTML5为开发语言，CSS3技术编写页面样式，Zepto.js技术来实现动态HTML页面以及用户与界面的交互，后端接口使用基于Node.js平台的Koa框架跟MongoDB来开发，其中还用了Vue.js来将接口数据渲染进去DOM。

### 2.2.1 HTML5

超文本标记语言第五代版本，增加了语义化标签，废除一些不好用的属性，同时还多了很多好用的新语法特性，比如本文会用到的本地缓存技术localStorage，还有支持多媒体的canvas，video，audio标记。HTML5很适合构建手机应用，最大的优势就在于可以在网页上调试跟修改。

### 2.2.2 CSS3

一个精美的网站少不了样式的装饰，如果把HTML5搭建出来的框架想象成一个屋子，那么CSS3就是这个屋子的装饰品，有了它，屋子才会变得漂亮。CSS3就是层叠样式表的简称，它可以控制网页布局，修改网页背景，字体大小以及控制其他页面表现。有了CSS3，前端开发者再也不用图片去制作小图标，因为CSS3有了圆角表格特性，比如本文用到的border-radius可以用来做圆角。以及新增的动画属性，变形(transform)、转换(transition)和动画(animation)，可以直接使用，让我们在不操作DOM结构的情况就实现了动画效果。

### 2.2.3 Zepto.js

轻量级的开源JavaScript库，本系统选用了该库，因为文件比较小巧，仅有8K多，在目前功能完备的库里面算是最小的，所以比起jQuery更适合移动端开发，兼容jQuery，让习惯了使用jQuery的开发者在不用耗费精力学习API的同时又能获得更快速度。

### 2.2.4 Vue.js

时下很火的用户界面框架，关注视图层，容易上手，支持组件化开发，通过尽可能简单的API实现数据绑定，让开发者不用去操作DOM。

### 2.2.5 Node.js

简单来说它就是运行在服务端的JavaScript, 它不是JavaScript应用，而是JavaScript平台，是一个基于Google的V8引擎的事件驱动I/O服务端JavaScript环境， 轻量且高效，为前端开发者带来便利，可以用熟悉的JavaScript语言去做web服务，使得前后端语言风格统一。

### 2.2.6 Koa1.0

基于node.js的web开发框架，是Express框架的人员开发出来的，可以用于快速搭建项目以及异步编程，比起Express更加轻量级，没有封装过多的中间件，让开发者只选择自己最需要的中间件去开发。在Koa的中间件中最重要的是generator函数，它是在请求基础上，以一个“后进先出”的模式去执行的。本次使用的是Koa1.0版本，如今已经出现了Koa2.0,跟1.0不一样的地方就在于语法的区别，本系统选用的1.0版本。Koa中有四个重要概念：Application,Context,Request,Response.

|  |  |  |
| --- | --- | --- |
| **Koa概念** | **说明** | **解释** |
| Application | 应用 | 开发过程以Koa应用开始，使用var app = new koa()创建应用 |
| Context | 上下文 | Koa的context将node的请求与响应封装在一个单独对象中 |
| Request | 请求 | 对node的request做进一步封装，提供了http服务的一些便捷有用的方法 |
| Response | 响应 | 对node的response做进一步封装，提供了http服务的一些便捷有用的方法 |

表1. Koa概念说明

### 2.2.7 MongoDB

著名的NoSQL数据库，文件存储类型为BOSN（JSON的扩展），以键值对的形式存储。在对比了几个数据库后，发现MongoDB更适合用于配合Node.js来进行开发，操作方便，性能高。

#### 1）NoSQL分类

NoSQL是指非关系型的数据库，适合于大量数据的存储，其中本系统所选用的MongoDB为文档存储类型的数据库，适合存储书籍内容。



图1. NoSQL分类图

#### 2）MongoDB介绍

在MongoDB中，有些概念跟我们平时常见的关系型数据库中的不太一样，在此说明一下。

|  |  |  |  |
| --- | --- | --- | --- |
| **SQL术语** | **解释说明** | **MongoDB术语** | **解释说明** |
| database | 数据库 | database | 数据库 |
| table | 数据库表 | collection | 集合 |
| row | 数据库记录行 | document | 文档 |
| column | 数据库字段 | field | 域 |
| index | 索引 | index | 索引 |
| table joins | 数据库表连接 | / | 不支持 |
| primary key | 主键 | primary key | MongoDB会自动将\_id字段设置为主键 |

表2. MongoDB与SQL相关概念比较表

## 2.3 论文组织结构简介

论文主要组织结构由七个部分组成，如下所示。

### 2.3.1 绪论

第一章的绪论简述了项目开发的背景以及当今的国内外现状。

### 2.3.2 概述

第二章概述部分主要介绍了课题概述以及开发本系统所涉及的技术。

### 2.3.3 系统分析

第三章系统分析主要针对系统进行可行性分析，开发平台简介以及需求分析。

### 2.3.4 系统设计

第四章系统设计从总体设计目标开始，通过直观的系统功能模块图，业务流程图，数据字典表格，数据流图来介绍系统完整的设计流程。

### 2.3.5 系统实现方案

第五章系统实现方案详细介绍了整个系统从服务搭建开始到整个系统完成提交的每个步骤，附上相关代码片段，代码注释，以及截图。

### 2.3.6 系统核心功能实现

第六章系统核心功能实现介绍了系统中五个核心功能的实现方法，包括轮播图，本地存储功能，注册功能，渲染数据功能以及翻页操作的具体实现。

### 2.3.7 系统测试

第七章系统测试从测试目的，测试对象，测试流程以及测试结论四方面对本项目进行运行测试，包含测试过程的系统截图。

### 2.3.8 结束语

第八章结束语对整个系统进行总结。

# 3.系统分析

## 3.1 可行性分析

在产品开发初期，对产品进行可行性分析是一件重要的事情，可以保证产品的可行性。在该系统中，我们主要从三个方面对系统进行可行性分析。目的是为了在开发过程中可以有明确且可行的方案。

可行性分析包括两个部分，从市场需求跟资源供应方面来看，由前面的分析可得知现在开发一款移动web电子阅读器，符合大众普遍要求。资源方面，现在互联网时代，资源共享，该系统的资源大部分可以从互联网上获取，放进去自己的项目里面，方便用户随时阅读。Web App跨平台，实现“一次开发，到处运行”，从各个方面来看，我们开发的这个系统是对用户而言有着积极的作用。

### 3.1.1 技术可行性

由前文列出的技术来看，HTML5跟CSS3新增的元素和各种属性，使得可以代替Flash动画功能，同时由于语义化标签的加入，代码的可读性也更强，更利于搜索引擎优化。在性能方面，从全球的几大主流站点都使用了HTML5来开发可以看出，它值得使用。后端接口开发使用node.js框架—Koa，对于那些偏向于前端开发的程序员来说是一件好事，可以使用自己熟悉JavaScript语言来的编写服务器端代码，交给Node.js去解释执行。因为，在技术上来看，使用主流的HTML5+CSS3，加上易于上手的Node.js是可行的。

### 3.1.2 经济可行性

开发一个Web App成本不高，无需购买服务器，后端接口可以使用开源框架koa以及免费的MongoDB，前端界面搭建，只有有一台笔记本，在任何操作系统下，随时随地可以进行开发。此处推荐使用Sublime Text3,轻量级，可以提高开发效率。调试程序可以使用谷歌浏览器自带开发者调试工具，功能强大，简单按下F12即可进行调试。从各个方面看，开发一个移动web电子阅读器几乎不用任何成本。

### 3.1.3社会可行性

社会可行性是指产品在社会中所起到的影响，主要包括以下三个方面。

#### 1）道德可行性

从道德上来讲，本系统没有涉及传播社会负能量，符合道德标准以及大众审美。

#### 2）法律可行性

本系统所涉及的书籍内容均来自于互联网上开放资源，在资源共享时代，该系统的书籍内容资源均可以从互联网上获取，在书籍内容的引入方面，会对资源内容先行进行审核，防止出现不良内容。

#### 3）用户使用可行性

本系统开发的是一款移动web电子阅读器,由于web app跨平台的优点，对用户的使用平台几乎没有要求，在适配方面本系统会针对不同分辨率手机做相应的适配，确保用户浏览时界面友好。由前面的分析可得知开发本系统，可以符合大众普遍要求。

## 3.2 开发环境

### 3.2.1 硬件平台

选择一个正确的硬件平台对于软件开发十分重要。不仅可以让我们快速开发项目，也可以更好地与软件结合起来，提高用户体验，让我们所开发的项目更好地展现出来。开发中，我们会选择性价比最高的平台来进行开发，本系统所选用的硬件平台为我们平常所使用的笔记本电脑，配置为普通标准配置，由于本系统使用基于HTML5+CSS3+Zepto.js开发，普通标准的硬件设备足以开发需求，无需另外购买高配置的硬件平台。其中本次开发所使用的硬件平台标准配置如下所示：

系统用户的硬件配置：Intel 2.2G 处理器，2G内存，320G 磁盘存储空间；

系统管理者的硬件配置：Intel 2.5G 处理器，4G内存，500G 磁盘存储空间。

### 3.2.2 软件平台

软件平台是指完成一个系统所需要的软件环境，包括客户端，服务端以及代码编写所需要的软件。正确合理地选择开发软件可以加快开发速度，在结合当下流行且实用的工具中，本系统最终确定选用以下的软件平台：

#### 1）操作系统

任何软件都必须在操作系统下才能运行。在微软发布的windows系统中，如今最新的是已经win10系统，本次开发是使用win7 64位操作系统，win7系统历时比较久，系统更加稳定，大部分用户在尝试了新发布的系统后仍然选用win7.

#### 2）开发软件

开发软件，顾名思义就是开发者进行系统开发时所使用到的软件一系列的软件和工具。对程序员而言，开发软件一般包括了脚本编辑器、服务器、数据库管理器、以及测试调试工具。本系统的开发软件列表如下：

a)脚本编辑器：

Sublime Text3。近年来大部分企业开发都使用这个工具。虽然是收费软件，但是可以无限次试用，黑色的用户界面，更加护眼，强大的功能，可以自定义快捷键；同时又错误提示，支持代码高亮。使用快捷键打开文件以及查找代码段方便。右侧还设有代码文件缩略图滚动条。方便快捷，值得使用。在本次系统开发过程所使用的编辑器中，按“Ctrl+F”可以快速查找代码，“ Ctrl+P”可以快速打开文件，“Ctrl+Shift+R”可以一键将代码调整至标准格式。

b)调试工具:

代码调试时每个程序成功运行的必经之路，也是每个程序员都必须苦苦挣扎的一个点。在没有借助任何工具的情况下，调试代码不是一件容易的事情，开发者无法定位代码出错位置，这样子只能盲目查找代码。如今，好在Chrome浏览器自带了功能强大的开发者调试工具，Elements栏目查看以及测试修改页面样式，Console栏目可以打印输出以及测试，Sources栏目可以查看项目源码，Network可以查看页面资源请求情况，还有其他强大的功能，比如设置断点调试。作为前端开发者，熟悉掌握调试技能也是性能优化的必备条件。Chrome浏览器还有自带手机客户端模拟器，可以让开发者轻易看出项目在不同分辨率手机中的显示情况。



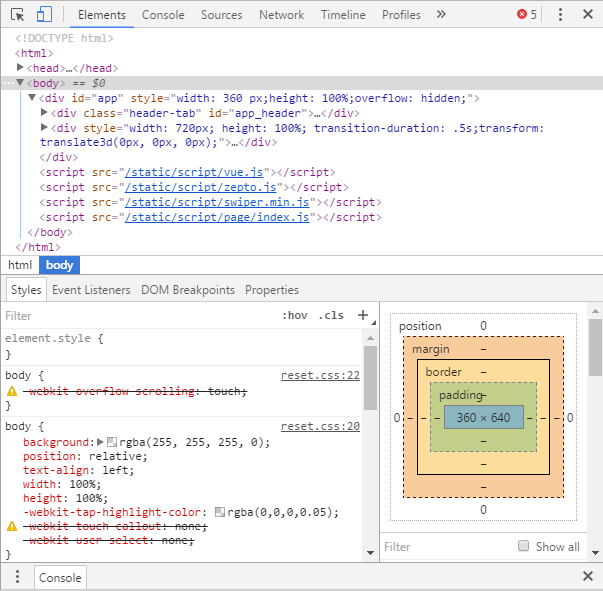
图2. Chrome自带调试工具模拟不同设备截图

图3. Chrome自带调试工具截图

## 3.3 需求分析

需求分析，通俗来说就是指我们在做一件事情之前先对事情进行分析，明白这件事情需要我们做的地方以及这件事情要完成什么功能。因此在系统开发前，对系统进行需求分析，一来是为了开发出实用性高的系统，二来是为了减少在软件开发过程中反复修改需求而造成的开发成本与开发精力的浪费。在企业的开发过程中，需求分析通常由产品经理来完成，产品经理先经过前期调研，对用户的需求不断进行调查以及研究，再根据类似产品去进行竞品分析，最终确定产品需求，撰写一份需求文档。在本系统开发之前，我们也对市场上先有的阅读器进行分析，调查用户最喜欢的哪些功能，以及有哪部分功能需要改进，最终确定了我们所要开发的电子阅读器包含以下内容。

本系统由分为三个部分开发：先对阅读小说的“阅读器”进行功能需求分析，看了市面上流行的大部分阅读器，总结发现阅读器需要具备三个基本功能，白天/夜间模式切换（夜间可设置护眼模式，提高用户体验），设置字号大小（用户阅读小说时会跟阅读器长时间接触，不同用户对字体大小要求会有不同），翻页按钮（方便用户使用）；有了阅读器，我们需要有一个入口可以进去阅读器，于是我们需要一个书籍列表展示界面，即书城界面；我们还需要一个管理员界面，来对我们书籍进行编辑管理，以下是我们对本系统需求的详细分析。

### 3.3.1 阅读器开发需求

在对一个项目进行需求分析之后，我们使用“墨刀”工具画出下面的设计原型稿，再继续进行需求详解。

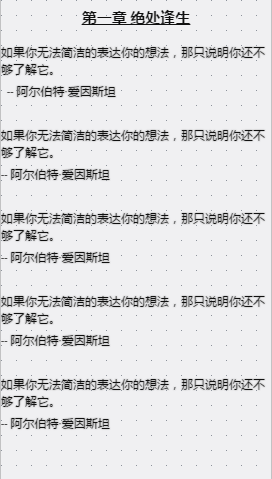
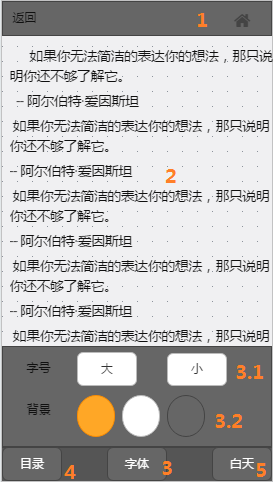
 

图4. 阅读器设计原型稿 图5. 阅读器字体操作面板设计原型稿

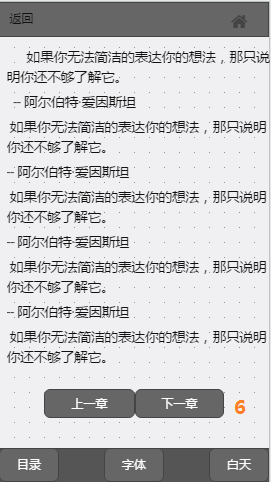
 

图6. 上下按钮设计原型稿 图7. 目录页设计原型稿

#### 1）公共返回导航

在搭建一个页面时，总会从总到分，从头到尾去写结构。此处我们设置一个公共的导航条用于返回上一级界面，这个功能的方便用户随时返回首页。

#### 2）文章内容主体

阅读器的主体部分，用于展示小说内容，点击主体部分中间区域，即可唤出上下导航条以及设置面板，这种“ show/hide”设置对用户比较友好，在用户需要设置页面样式的时候再单击唤出，不需要的时候则隐藏起来。

#### 3）字体按钮

点击该按钮可以唤出“字号大小设置”以及“背景颜色设置”的面板，让用户根据自己需求调整阅读模式。

a)字号大小设置

考虑到不同用户对字体的要求会不一样，该功能用户可以根据自己的需要设置字体大小，本系统对页面字体大小设置有上下限的限制，让用户界面达到最大的和谐度。同时要保证浏览器可以记忆设置好的选项，即刷新之后依然呈现用户最近一次设置好的字号大小。

b)背景颜色设置

分别针对白天有五个不同的背景颜色，以及针对夜间的一个黑色背景颜色。若用户想要改变背景颜色，单击颜色按钮即可将背景颜色调整至最舒适的状态。该按钮也有记忆用户上一次设置的功能。

#### 4）目录按钮

点击目录按钮会跳转至本书相应的目录详情页。

#### 5）白天/夜间模式切换按钮

白天模式默认的背景颜色为浅棕色（#e9dfc7），夜间模式的背景颜色为黑色（#000）；用户点击该按钮即可进行模式切换。

#### 6）上一章/下一章翻页按钮

点击“上一章”或“下一章”按钮可以对小说进行翻页显示。

#### 7）目录详情页

目录详情页展示了该书籍的具体章节信息。

### 3.3.2 书城展示列表需求

#### 1）网站header

书城网站采用水平侧滑的形式展示了两个一级页面，左右两边包括“签到”入口和“登录”入口，其中登录入口包含两个按钮：“注册”跟“游客”模式。

a)用户登录

使用注册后的帐号登录书城系统，若没有帐号，先进行注册，若用户不想登录，可点击“游客”按钮浏览书城。

b)用户注册

注册信息包括用户名跟密码，确认密码，用户名为手机号码，需要进行合法手机号码校验，密码跟确认密码一致才可以注册成功。

c)签到

“签到”读书功能，鼓励用户阅读，用户登录后，可以进行签到。

#### 2）书城顶部功能区域

作为该Web App顶部功能区域，包括三个部分：搜索框，活动页轮播图，分类栏目。

a)搜索框

输入“书名/作者/关键字”实现全局搜索，成为用户寻找书籍的快速接口。

b）活动页轮播图

作为众多网站必备的元素，本系统的轮播图使用Swiper**[10]**技术实现，它是目前应用比较广泛的移动端网页触摸内容滑动的JS插件。

c）分类栏目

分类设有“男生频道”，“女生频道”，“分类”，“排行”，点击可以进入二级详情页面。

#### 3）书籍列表模块

书籍列表模块展示了网站热门小说，分别有以下版块：“本周最火”，“重磅推荐”，“女生最爱”，“男生最爱”， “限时免费”，“精选专题”共六大模块，其中“女生最爱”与“男生最爱”每次展示五本书，底部设置“换一换”按钮可切换书籍。

#### 4）书架

书架用于展示用户选中的书本，在右侧的切换按钮可以将书架上的书籍列表切换为内联显示形式。

#### 5）书籍详情

书籍详情页展示了书籍详情以及作者的其他相关作品。

#### 6）搜索详情页

搜索详情页面显示有默认搜索关键词，在输入框键入关键词可以得到搜索结果，若搜索不到然后内容，则显示“没有相应的搜索结果”。

### 3.3.3 管理员后台开发

用户登录时，若为管理员账号，会跳转至管理员添加页，管理员有权限添加书籍以及删除书籍。

#### 1）管理员添加页

管理员可以添加书籍，包括书籍所属栏目的名称，书籍封面，书籍标题，书籍简介，书籍作者以及书籍标签。

#### 2）管理员编辑页

管理员可以查看已经添加的书籍信息详情，以及管理员拥有删除书籍信息的权限。

### 3.3.4 服务端接口开发

服务端接口由Node.js驱动，安装Koa框架用于快速搭建网站，其中服务器插件均使用Node.js自带的npm(Node.js包管理工具)安装。服务器接口主要包括首页接口，频道详情页数据接口，登录接口，注册接口，搜索接口以及管理员后台模块的接口。

### 3.3.5 系统性能需求

#### 1）设计标准要统一

项目开发过程中要求设计标准要统一。对功能模块进行合理地规划开发，保证每个功能模块可以独立运行有可以跟其他模块协作，即我们平时所听说的“高内聚，低耦合”**[11]**

#### 2）系统响应时间要求

a)服务端响应时间

服务端数据交互的响应时间需要控制在2秒钟之内，这样子确保系统响应快速，用户不会失去耐心。因此需要对数据库以及服务端接口进行合理设计。

b)前端交互响应时间

前端页面的交互时间同样应该控制在2秒钟之内，因此在开发过程应该对前端页面性能做到最大的优化。

#### 3）编码风格符合规范

想象一下自己去到一个新公司，老大拿出了一份前同事写完的代码，发现没有任何注释，页面缩进有问题，逻辑不清晰，这样子你需要花费大量时间精力去研究代码，大大降低了开发效率。因此，在我们项目开发过程中，我们应该要严格遵循代码规范原则编写代码，代码风格保持统一才容易被维护跟理解。对于每一个模块，应该使用注释进行解释说明，这样子在合作开发过程中才可以高效工作。

## 3.4 系统设计

系统设计是根据前面论述的所有需求，对系统进行从功能，业务流程，数据字典，数据流等方面的设计，是系统功能实现之前的必要准备工作。

### 3.4.1 总体设计目标

#### 1）前端界面友好

a)系统界面符合用户审美，色彩搭配合理，视觉效果好；

b）系统交互合理，符合用户日常习惯，易用性高，用户上手快；

c）系统界面可以适配市场上所有主流手机，在不同分辨率手机下可以友好显示，兼容性好；

#### 2）系统可用性强

a)用户点击界面时可以快速响应页面

b)系统采用“渐进增强”**[12]**方式开发，保证在未来的开发过程中进行功能追加。

### 3.4.2 系统功能描述

有了前面的需求分析，我们初步确定了本系统的功能以及技术选型，接下来我们便可以进行系统设计。

本系统为B/S结构应用，其中前端页面使用HTML5+CSS3+Zepto.js开发，服务端接口使用Node.js开发，接口主要为前端提供数据，本系统最重要的部分在于前端界面开发以及与用户交互部分的开发。

阅读器模块实现小说内容展示，阅读器背景颜色设置，字号大小设置，上下翻页功能以及点击返回首页等。

书城模块实现书籍展示，书籍搜索，签到，登录注册，点击进入书籍详情页等功能。

服务端接口实现了后端逻辑，实现数据库存储以及数据的增删改查。

### 3.4.3 系统功能模块图

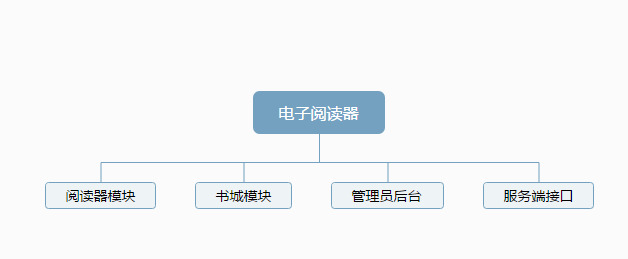


图8. 系统功能模块图

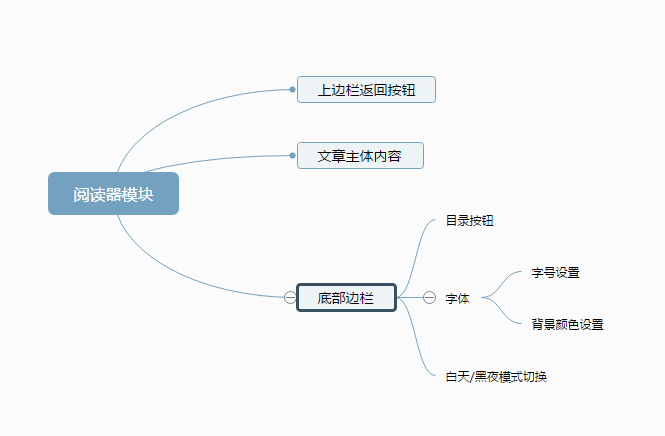


图9. 阅读器功能模块图

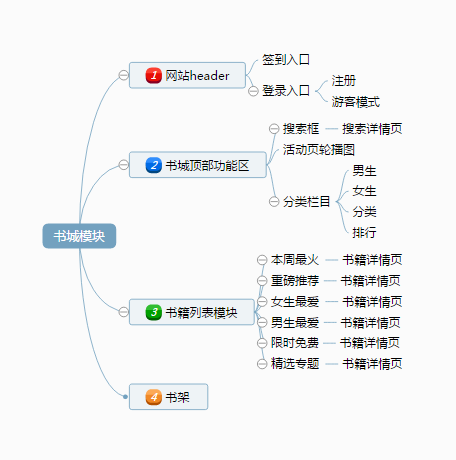


图10.书城功能模块图

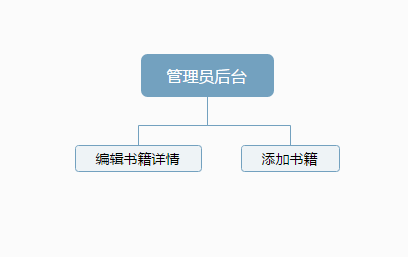


图11.管理员后台功能模块图

### 3.4.4 业务流程图

进入书城

登录

否

是

注册

游客

管理员

用户

添加书籍

编辑书籍

签到

浏览首页

查看书籍列表

点击查看书籍详情

开始阅读

操作阅读器

返回首页

结束阅读

图12. 系统全局业务流程图

### 3.4.5 数据字典

数据字典是根据业务流程中所需要的数据信息进行详细说明，在本系统中，根据需求分析以及业务流程，我们需要有记录首页书籍信息，男生频道书籍信息，女生频道书籍信息，书籍目录信息，排行榜书籍信息，书籍分类信息，书籍详情信息共八张表。由于本系统使用的是MongoDB，跟关系型数据库有所区别，集合相当于关系型数据库中的表，域相当于关系型数据库中的字段。

#### 1）首页书籍信息表

作用：用于记录展示在首页的所有书籍信息

表3. 首页书籍信息表

|  |  |
| --- | --- |
| 数据存储名： | 首页书籍信息 |
| 说明： | 记录首页所有书籍信息，包括书籍ID、书籍标题、书籍简介、书籍所属标签、书籍作者、书籍封面图片 |
| 数据描述： | 书籍ID+书籍标题+书籍简介+书籍所属标签+书籍作者+书籍封面图片URL |
| 集合名称 | homes |
| 域名称 | fiction\_id+title+summary+tags+authors+cover |

#### 2）男生频道书籍信息表

作用：用于记录男生频道展示列表的书籍信息

表4. 男生频道书籍信息表

|  |  |
| --- | --- |
| 数据存储名： | 男生频道书籍信息 |
| 说明： | 记录男生频道所展示的书籍信息，包括频道分类、书籍标题、书籍简介、书籍所属标签、书籍作者、书籍封面图片 |
| 数据描述： | 频道分类+书籍标题+书籍简介+书籍所属标签+书籍作者+书籍封面图片URL |
| 集合名称 | males |
| 域名称 | ad\_name+title+summary+tags+authors+cover |

#### 3）女生频道书籍信息表

作用：用于记录女生频道展示列表的书籍信息

表5. 女生频道书籍信息表

|  |  |
| --- | --- |
| 数据存储名： | 女生频道书籍信息 |
| 说明： | 记录女生频道所展示的书籍信息，包括频道分类、书籍标题、书籍简介、书籍所属标签、书籍作者、书籍封面图片 |
| 数据描述： | 频道分类+书籍标题+书籍简介+书籍所属标签+书籍作者+书籍封面图片URL |
| 集合名称 | females |
| 域名称 | ad\_name+title+summary+tags+authors+cover |

#### 4）书籍目录信息表

作用：用于记录书籍章节目录信息

表6. 书籍目录信息表

|  |  |
| --- | --- |
| 数据存储名： | 书籍章节目录信息 |
| 说明： | 记录书籍详细目录信息，包括书籍章节目录信息以及本章节是否免费的信息 |
| 数据描述： | 书籍章节目录信息+是否免费信息 |
| 集合名称 | catedetails |
| 域名称 | name+free |

#### 5）排行榜书籍信息表

作用：用于记录排行榜书籍信息

表7. 排行榜书籍信息

|  |  |
| --- | --- |
| 数据存储名： | 排行榜书籍信息 |
| 说明： | 记录排行榜书籍信息，包括排行榜封面以及排行榜书籍名称 |
| 数据描述： | 排行榜封面+排行榜书籍名称 |
| 集合名称 | ranks |
| 域名称 | cover+  description |

#### 6）书籍分类信息表

作用：用于记录书籍的分类信息

表8. 书籍分类信息表

|  |  |
| --- | --- |
| 数据存储名： | 书籍分类信息 |
| 说明： | 记录书籍分类信息，包括分类封面图，分类名称，每个分类的书本数目 |
| 数据描述： | 分类封面图URL+分类名称+所属分类书本数目 |
| 集合名称 | categories |
| 域名称 | new\_image+label+fiction\_count |

#### 7）书籍详情信息表

作用：主要记录书籍详情信息

表9.书籍详情信息表

|  |  |
| --- | --- |
| 数据存储名： | 书籍详情信息 |
| 说明： | 记录书籍详情信息，包括书籍ID，书籍字数，书籍评价数，书籍名称，书籍内容，书籍价格，书籍作者，书籍版权，书籍封面URL，最新章节，作者的相关书籍以及喜欢本书的人也喜欢的书籍，书籍所属标签等 |
| 数据描述： | 书籍ID+书籍字数+书籍评价数+书籍名称+书籍内容+书籍价格+书籍作者+书籍版权+书籍方面URL+最新章节+作者的相关书籍名称+相关书籍封面URL+喜欢本书的人也喜欢的书籍名称+喜欢的书籍封面URL+书籍所属标签 |
| 集合名称 | books |
| 域名称 | book\_id+word\_count+score\_count+title+content+price+authors+rights+cover+latest+related\_title+related\_cover+author\_title+author\_cover |

#### 8）用户信息表

作用：主要记录注册后的用户信息表

表10.用户信息表

|  |  |
| --- | --- |
| 数据存储名： | 用户信息 |
| 说明： | 记录注册后的信息，包括用户名，密码 |
| 数据描述： | 用户名+密码 |
| 集合名称 | users |
| 域名称 | username+password |

### 3.4.6 数据流图

移动web电子阅读器中的数据流存在于客户端，服务端以及数据库中，在本系统中，数据在浏览器，Koa服务以及MongoDB之间流动。

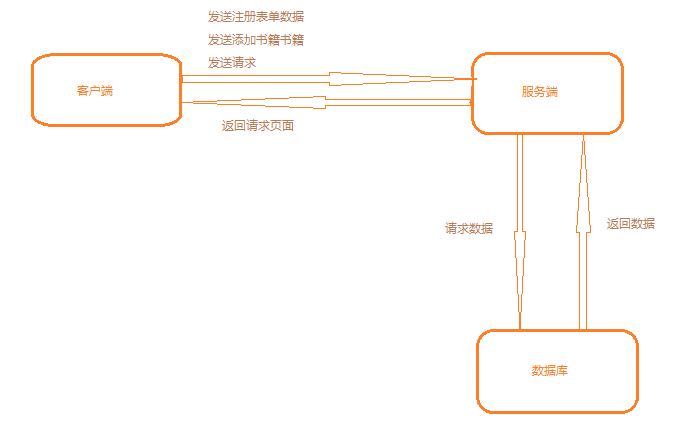


图13.系统数据流图

在这里我们介绍下本系统中管理员后台数据是如何流动的。当点击添加数据时候，进入管理员添加书籍页面，管理员填写表单，此时数据就从客户端开始了，点击“添加”按钮，此时，表单以post方式将数据提交到Koa所搭建的服务端中；当使用AJAX技术向服务端请求数据时，服务端会从数据库请求数据，接着数据库返回数据，服务端再将数据拼接起来返回到客户端中。

# 系统实现步骤

## 4.1 开发环境准备

系统开发前我们先准备好本系统开发所需要的硬件和软件条件：

硬件：一台win7 64位笔记本电脑；

软件：Chrome浏览器:用于Web 电子阅读器的展示，模拟手机以及调试代码；

Sublime Text 3:作为本系统编码主要工具，提供各种高效率插件以及快捷键；

Git Bash:版本控制工具Git在Windows下的命令行工具，用于项目版本控制，确保开发过程代码的更新以及维护；

Node.js：JavaScript运行环境，用于便捷地搭建响应速度快、易于扩展的Web应用，在开始项目之前需要先安装好该环境，可以采用全局安装的方式；

MongoDB：高性能的数据存储工具，用于该项目的数据存储；

Robomongo: MongoDB的可视化工具，方便录入测试数据。



图14. 项目开发使用的软件图

## 4.2 开始项目

### 4.2.1项目文件搭建

新建项目文件夹webAppReaderDemo，在其下新建database文件夹用于存放跟数据库相关文件，mock文件夹用于存放调试过程的json模拟数据文件，node\_modules为安装项目依赖时自动生成的文件夹，存放着各个依赖文件，service文件夹用于存放服务器端接口文件，static为静态资源文件夹，细分为css文件夹—用于存放项目样式文件，img文件--用于存放项目所需图片资源，script文件夹—用于存放项目的所有JavaScript文件，view文件夹用于存放所有视图文件，app.js作为整个项目的入口文件，package.json为项目依赖配置文件。

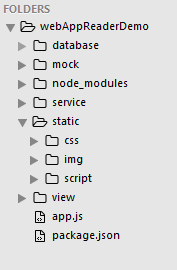


图15. 项目骨架图

### 4.2.2项目依赖初始化

在package.json写下如下配置：

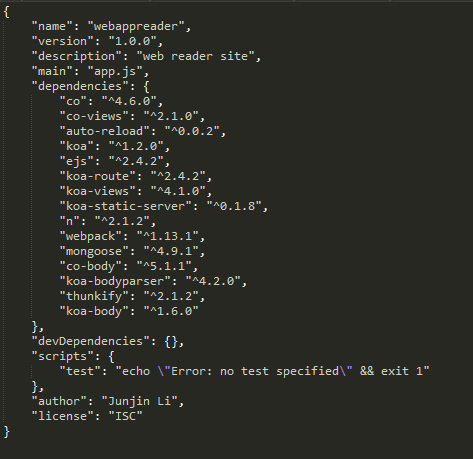


图16. 项目依赖截图

在项目开始前我们已经安装好Node.js环境，我们首先使用windows命令行控制工具，进入webAppReaderDemo所在的文件目录，在该目录下使用"npm install”命令行安装所有项目依赖模块，npm**[8]**是一个包管理工具，开发人员可以根据自己需要去安装模块。以上所列出的模块就是本系统所需的全部模块，本系统为单页面应用加路由页面，所以我们引入了路由模块”koa-route”来实现路由，”ejs”为页面模板引擎，用于渲染页面，”mongoose”为MongoDB快速建模工具，”koa-static-server”是用于集成项目静态资源,”co-body”用于处理页面提交表单时，将表单内容转化为对象形式。

### 4.2.3项目入口文件app.js

var koa = require('koa'); //koa中间件

var controller = require('koa-route');

var app = new koa(); //渲染模板

var views = require('co-views');

var render = views('./view', { //view是存放路径

map: {html: 'ejs'} //模板引擎

});

app.listen(3005);

console.log("Koa server is started!");

项目端口为3005，在项目目录下运行命令行”node app.js”,若koa服务开启成功，看到命令行控制台打印输出” Koa server is started!”。

### 4.2.4 创建视图层文件

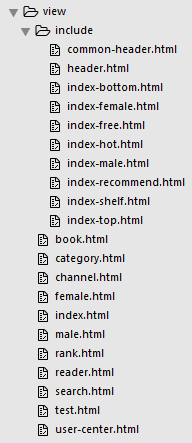
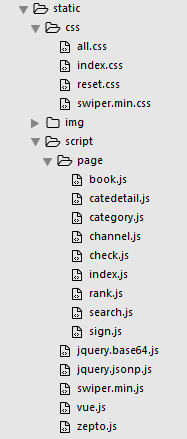
 

图17. 视图层文件 图18. 项目文件

根据功能需求分析，因为本系统为单页面应用，书城主页面为index.html,其他均为路由页面，首页结构相对复杂，我们使用组件化方式开发，首页组件存放在include文件夹中，开发过程使用ejs模板的语句即可引入，如下所示：

<div class="container-scroll" style="top: 0px;">

<% include include/index-top.html%>

<% include include/index-hot.html%>

<% include include/index-recommend.html%>

<% include include/index-female.html%>

<% include include/index-male.html%>

<% include include/index-free.html%>

<% include include/index-bottom.html%>

</div>

阅读器开发为单独一整个模块，写在reader.html中，等开发完毕再集成到项目中。

页面样式开发方面，为了达到性能最佳，减少静态资源请求次数，我们将具备相同样式的元素使用同个类名，集成样式，这样子可以减少代码冗余，其中，把首页所有样式写进”index.css”,其他页面的样式写进“all.css”，”reset.css”用于重置HTML标签在浏览器里面的默认样式。

在script文件夹中有pages文件夹用于存放页面相关的js文件，其余的为开源js库。

jquery.base64.js插件：用于将base64格式的数据解码，通常我们为了安全起见，我们所请求的线上数据是被以base64方式加密处理过的，我们接收到数据后需要进行解密才可以正常显示在页面中。

jquery.jsonp.js插件：可以实现跨域数据处理。

//数据解密过程需要用到以上两个插件

function getBSONP(url, callback) {

return $.jsonp({

url : url,

cache : true,

callback : "duokan\_fiction\_chapter",

success : function(result) {

var data = $.base64.decode(result);

var json = decodeURIComponent(escape(data));

callback(json);

}

});

};

### 4.2.5 页面路由开发及测试前端流程

开发过程中随时对页面进行测试是一件必要且重要的事情，在我们还没将全部页面开发完成时，我们需要随时对页面进行测试。页面路由添加方式如下，在app.js文件里添加如下代码，其他页面路由也按照以下方式添加：

app.use(controller.get('/', function\*(){ //默认访问到首页

this.set('Cache-Control', 'no-cache');

this.body = yield render('index', {title: '书城首页'});

}));

app.use(controller.get('/search', function\*(){

this.set('Cache-Control', 'no-cache');

this.body = yield render('search', {title: '搜索', nav: '搜索'});

}));

在index在浏览器输入“http://127.0.0.1:3005/ ”，若能访问到首页文件，说明页面打通了。

### 4.2.6 设计数据库模型

使用MongoDB的快速建模工具Mongoose，先连接到我们的本地数据库maplebookshop，在app.js文件夹里面创建连接数据库方式如下：

var mongoose = require('mongoose');

var db = mongoose.connect('mongodb://localhost/maplebookshop');//连接数据库

db.connection.on("error", function (error) { console.log("数据库连接失败：" + error); });

db.connection.on("open", function () { console.log("------数据库连接成功！------"); });

在Mongoose中有模式Schema,模型Model,文档Documents三个概念，建模过程如下：

/\*User表start\*/

//创建模式（相当于骨架），但是该模式还未和user集合关联

var Schema = mongoose.Schema;

var userSchema = new Schema({

name: String,

password: String

});

//创建User模型

var User = mongoose.model('user', userSchema);

//创建文档实体

var admin = new User({

name: "lijunjin",

password: "maple"

});

//保存文档实体

admin.save();

/\*User表end\*/

### 4.2.7 在MongoDB客户端查看数据

为了确认数据是否被成功存储至数据库，在MongoDB客户端键入如下命令行即可查看数据，由下可见数据已被成功保存至数据库：

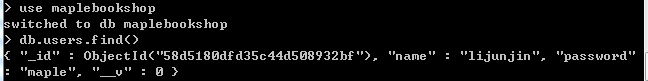


图19. MongoDB客户端运行情况图

也可以使用MongoDB可视化工具RoboMongo来查看数据，更加方便直观：

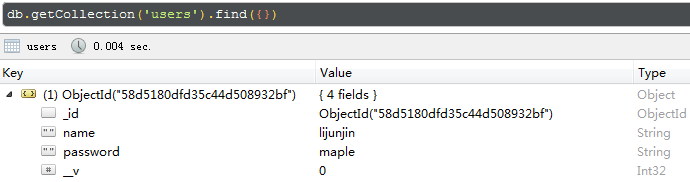


图20. RoboMongo运行情况图

### 4.2.8 后端业务逻辑开发

在本系统中，将服务端逻辑部分代码写在service中的webAppService.js文件以及app.js中：

//webAppService.js文件的代码是开发过程中用于读取模拟数据来测试页面的

var fs = require('fs');

exports.get\_test\_data = function(){

var content = fs.readFileSync('./mock/test.json', 'utf-8');

return content;

}

//首页数据

exports.get\_index\_data = function(){

var content = fs.readFileSync('./mock/home.json', 'utf-8');

return content;

}

//在app.js中采用require方法讲webAppService.js文件引入，并且在对应的路由中展示数据

var service = require('./service/webAppService.js');

app.use(controller.get('/ajax/index', function\*(){

this.set('Cache-Control', 'no-cache');

this.body = service.get\_index\_data();

}));

### 4.2.9 页面自适应屏幕宽度代码改造

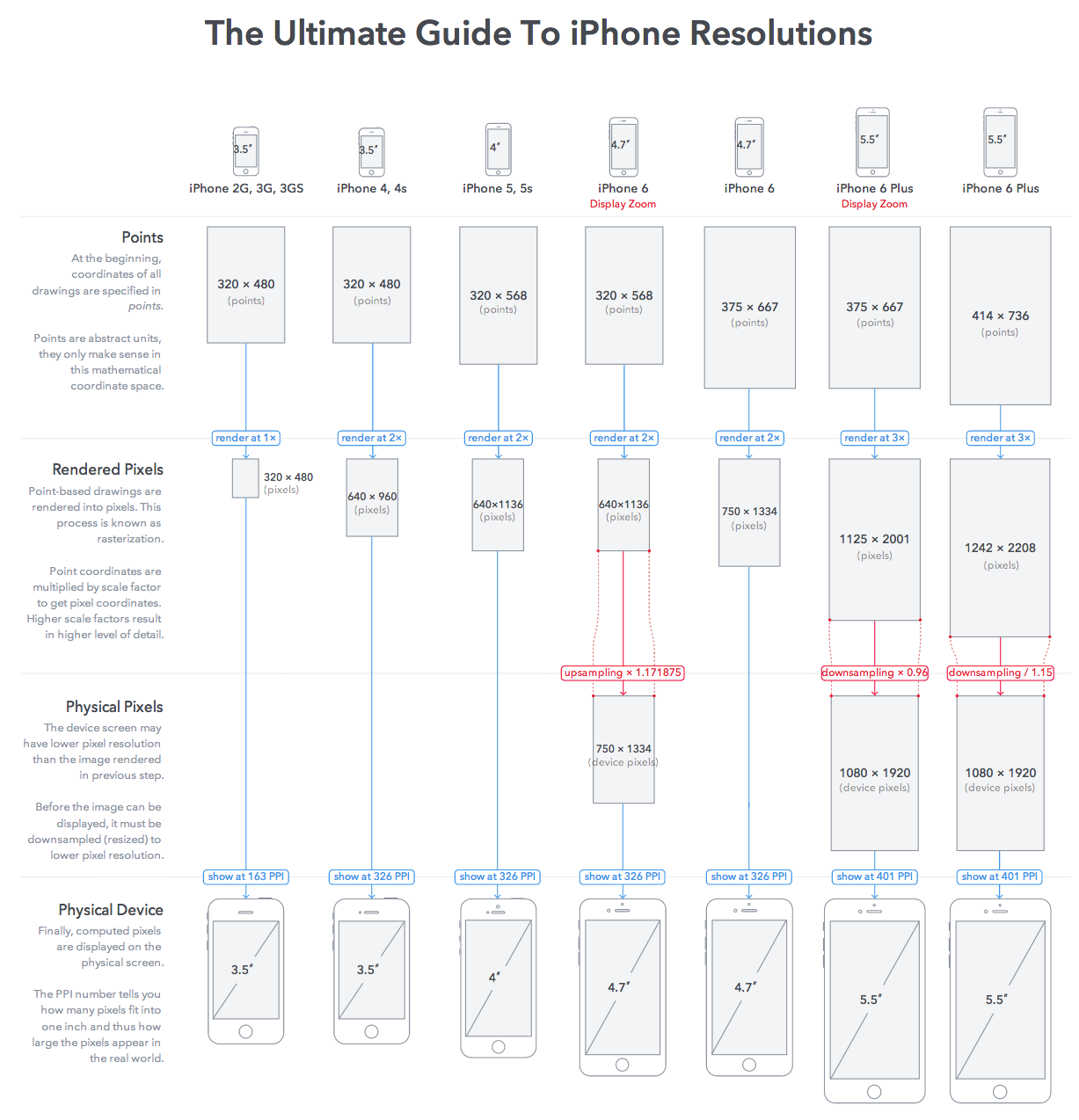


图21.iPhone屏幕宽度图

完成页面后需要对页面的功能以及样式进行调试，上图为网上出现的移动端需要适配的设备，由于移动端设备多，屏幕分辨率差距大，因此我们需要对页面做响应式处理，在本项目中我们对首页header做自适应调整，如下所示：

（1）对于移动端页面，都需要加上下面这句话，把布局视口的宽度改为屏幕的宽度，即理想视口**[9]**，这样让用户进入页面的时候无需缩放即可达到最佳浏览效果：

<meta name="viewport" content="width=device-width,initial-scale=1,user-scalable=no">

对于页面中需要自适应宽度的地方，采用自动获取屏幕宽度的形式：

<div style="width: {{ double\_screen\_width }}px; height: 100%; transition-duration: .5s;transform: translate3d({{ position }}px, 0px, 0px);">

<div class="container-wrap" style="width: {{ screen\_width }}px; height: 100%; float: left;"></div>

</div>

JS获取屏幕宽度代码如下，用Vue.js将数据渲染至视图层：

var windowWidth = $(window).width();

if (windowWidth < 320) {

windowWidth = 320;

}

new Vue({

el: '#app',

data: {

screen\_width: windowWidth,

double\_screen\_width: windowWidth\*2,

position: 0,

}

### 4.2.10 项目托管

在项目开发的全过程中，使用Git版本控制工具以及GitHub代码托管平台来实现代码的版本更新以及托管项目代码。

在Git本地目录下使用两行命令即可将项目提交至本地仓库

$ git add {fileName}

$ git commit –m ‘you can write down your project version description’

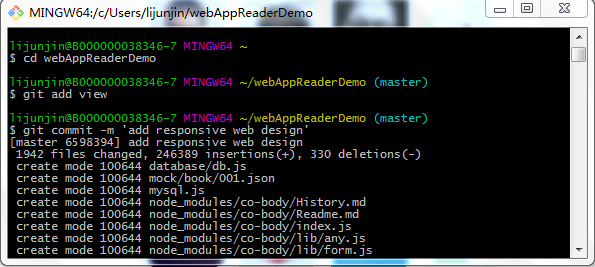


图22.git Bash界面截图

在项目被成功提交至本地库后，用push命令讲项目推送至GitHub远程仓库。

$git push origin master

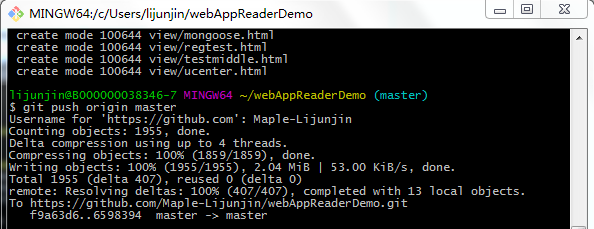


图23.将代码推送至远程

在GitHub上可以看到我们项目代码以及项目开发过程中代码提交记录。

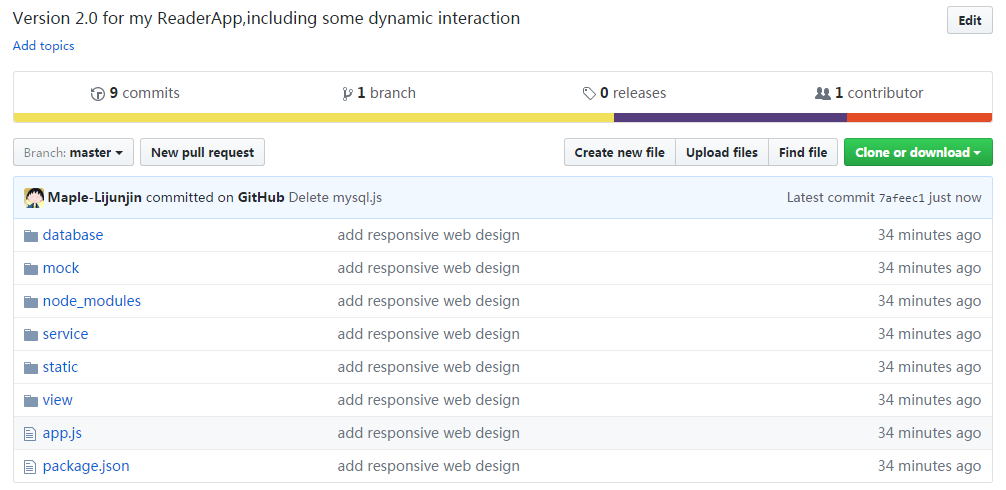


图24.GitHub项目代码图

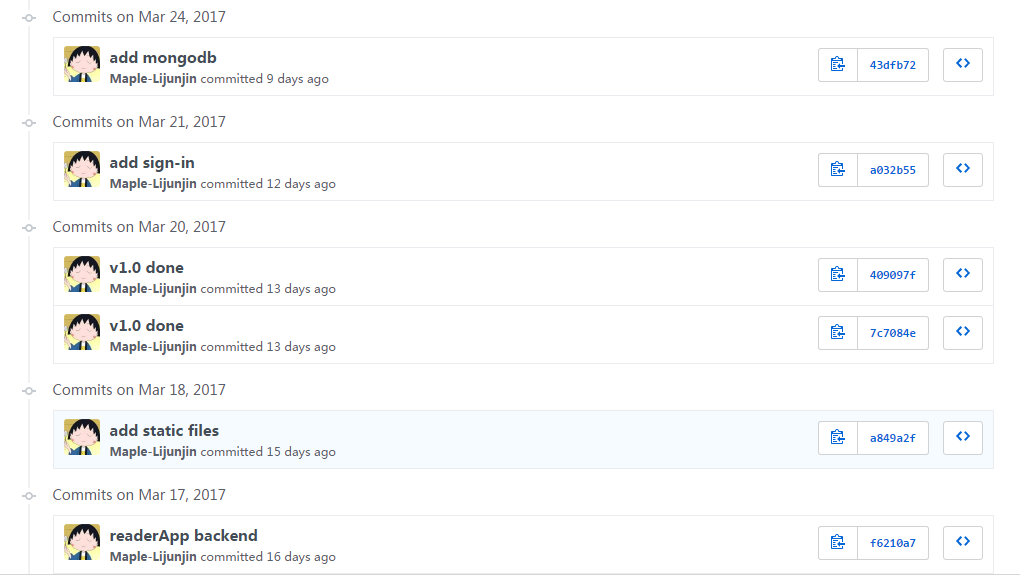


图24.GitHub项目提交记录图

# 系统核心功能实现方法

## 5.1 首页轮播图实现

首页轮播图在页面上可以吸引用户眼球，做活动宣传的效果，如今大多数网站在首页顶部都设有轮播图。

在本项目中，轮播图采用Swiper插件，它是使用纯JavaScript实现的页面滑动插件，使用简单，功能强大，可以根据不同的属性展现出不同的效果。

**实现思路：**

为轮播图加上Swiper规定的类名，按照Swiper的代码写法，定义轮播图大小，轮播图滑动时间间隔。

实现轮播图代码如下：

<html>

<head>

<link rel="stylesheet" type="text/css" href="/static/css/swiper.min.css">

</head>

<body>

<div class="swiper-container">

<div class="swiper-wrapper">

<div class="swiper-slide">

<a> <img class="cover" src="static/img/img1 " /> </a>

</div>

<div class="swiper-slide">

<a> <img class="cover" src=" static/img/img2"> </a>

</div>

<div class="swiper-slide">

<a> <img class="cover" src=" static/img/img1"> </a>

</div>

</div>

</div>

<script src="/static/script/swiper.min.js"></script>

<script>

var mySwiper = new Swiper ('.swiper-container', {

autoplay: 3000,

loop: true

});

</script>

</body>

</html>

**代码注释：**

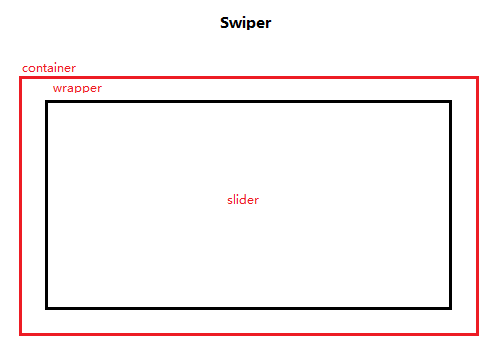


图25.Swiper插件结构图

使用Swiper插件，项目的页面层次如上所示，先把页面布局写好，然后在JS代码中新建一个Swiper对象，根据项目需要使用Swiper对象的属性，“autoplay：3000”表示轮播图自动播放，间隔时间为3000毫秒，“loop: true”表示图片循环播放。

## 5.2 字体大小及背景颜色记忆功能

为了提高用户体验，让阅读器拥有记忆功能，在用户每次打开阅读器都可以记住上一次的选项，我们使用了HTML5新特性localStorage，它是没有时间限制的本地存储，用户在关闭浏览器，长时间之后重新打开仍可看到最近一次的设置。但值得注意的是，本地存储是在本浏览器内起作用的，跨浏览器是获取不到的。

**实现思路：**

先将用户的设置存储到localStorage中，下一次使用的时候再到localStorage中获取。

浏览器记忆功能核心代码如下：

var StorageGetter = function(key) {

return localStorage.getItem(prefix + key);

}

var StorageSetter = function(key, val) {

return localStorage.setItem(prefix + key, val);

}

//字体放大

$('#large-font').click(function() {

if (InitFontSize > 20) {

return;

}

InitFontSize += 1;

Util.StorageSetter('font\_size', InitFontSize);

RootContainer.css('font-size', InitFontSize);

});

//字体缩小

$('#small-font').click(function() {

if (InitFontSize < 12) {

return;

}

InitFontSize -= 1;

Util.StorageSetter('font\_size', InitFontSize);

RootContainer.css('font-size', InitFontSize);

});

//常规字体设置信息

InitFontSize = Util.StorageGetter('font\_size');

InitFontSize = parseInt(InitFontSize);

if (!InitFontSize) {

InitFontSize = 18;

}

RootContainer.css('font-size', InitFontSize);

**代码注释：**

先将localStorage中的setItem()与getItem()方法封装成StorageSetter()跟StorageGetter()方法，方便后面的代码使用，当放大或者缩小字体时，先设置一个临界值，防止字体变得过大或者过小，使用封装好的StorageSetter()方法记住用户这一次的设置，当下一次打开时阅读器时候，StorageGetter()方法会获取上一次设置的字体InitFontSize。

## 用户注册功能的实现

注册功能是一个拥有用户的系统中必有的一个功能，在本系统中，前端注册界面包含的用户信息有：用户名为手机号码，用户密码，确认密码以及注册按钮。服务端则需要对注册表单所提交的数据进行处理，存入数据库。

**实现思路：**

在HTML5中，表单使用<form>元素，提交方法为post，表单中的输入框及提交按钮都使用<input>元素，用户名输入框的<input>元素类型为text，用户名为手机号码，需要使用正则表达式对手机号码进行合法性校验，合法的手机号码是以1开头，第二位为3,4,5,8中的任意一个，接下来的为0-9中的任意数字。密码输入框的<input>元素类型为password，密码需要跟确认密码完全一致才可以进入下一步，确认注册按钮的<input>元素类型为submit。在服务端，当接收到用户提交的数据时，先使用co-body模块把数据转换位对象模式，再使用MongoDB中的sava()方法将取到的用户名跟密码存储至数据库中。

注册功能核心代码如下所示：

//注册表单

<form name="registerForm" class="register-form" method="post" action="/sign" onsubmit="return checkRegister()">

<p><input id="username" type="text" name="username" value="" placeholder="请输入您的手机号码" onblur="checkPhone()"></p>

<p class="pwd"><input id="password" type="password" name="password" placeholder="密码" onfocus="disapear()"></p>

<p class="pwd"><input id="confirmpwd" type="password" name="confirmpwd" placeholder="确认密码" onfocus="disapear()"></p>

<p class="hide" style="visibility: hidden;">两次输入密码不一样</p>

<p><input id="register" type="submit" name="submit" value="立即注册"></p> </form>

**代码解释：**

注册表单使用post方法，action指表单所提交到的服务端地址为"/sign"，"/sign"路由在本系统中对应登录页面，即当注册成功时会跳转至登录页面，<form>元素中onsubmit属性指当点击表单提交按钮时所执行的方法。在用户名输入框中，有一个onblur属性，指当鼠标在该输入框失去焦点时会执行校验手机号码的方法。在表单中有一个被隐藏的<p>标签为密码与确认密码不一致时的提示语，当前后密码不一致时该标签会显示出来，此时用户会去点击表单修改密码，在密码输入框中使用onfocus属性，即当光标移到密码框去修改密码时，执行disapear()方法让提示文本消失。

//前端校验用户输入信息合法性

function checkPhone() {

var username = $("#username").val();

if (!(/^1[3|4|5|8][0-9]\d{4,8}$/.test(username))) {

alert("不是完整的11位手机号或者正确的手机号前三位");

return false;

} else {

return true;

}

}

function checkRegister() {

var password = $("#password").val();

var confirmpwd = $("#confirmpwd").val();

if (password != confirmpwd) {

$(".hide").css("visibility", "visible");

return false;

} else {

alert("注册成功！");

return true;

}

}

function disapear() {

$(".hide").css("visibility", "hidden");

}

**代码解释：**

checkPhone()方法用于对用户所输入的手机号码进行校验，若不符合则出现弹框，提示用户所输入的"不是完整的11位手机号或者正确的手机号前三位"并且”return false”退出，若符合则进入下一步；checkRegister ()方法用于校验用户输入的密码跟确认密码是否完全一致，若

不一致，则在密码输入框下方显示“两次输入密码不一样”的提示语，若成功则弹出"注册成功！"的按钮，并跳转至登录页面。disapear()方法用于让前后密码不一致时的提示文本隐藏。

//服务端处理用户表单

var render = views('./view', { //view是存放路径

map: {html: 'ejs'} //模板引擎

});

var controller = require('koa-route');

var parse = require('co-body');

app.use(controller.post('/sign', function\*(){

this.set('Cache-Control', 'no-cache');

var item = yield parse(this);

var username = item.username;

var password = item.password;

var user = new User({

username: username,

password: password

});

user.save(function(err){

console.log('save status:'+err);

});

this.body = yield render('sign', {title: '登录'});

}));

**代码解释：**

引用co-body模块，在服务端使用koa-route页面路由中的post方法，item即为传输过来的表单数据，使用Mongoose中的save方法将用户名跟密码存储在users集合对应的username跟password域名中:

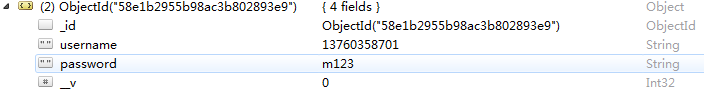


图26. 注册数据图

“this.body = yield render('sign', {title: '登录'});”将使用ejs模板将注册后会跳转过去的登录页面渲染出来。

## 5.4 后端数据渲染至前端的实现

一个完整的系统会有前端和服务端组成。通常情况下前端用于展示视图，后端用于实现跟数据库连接以及处理数据。本项目的数据存储在MondoDB的数据库中。

**实现思路：**

此处主要介绍一下“男生频道”前后端是如何打通的。本系统所用到的数据存储于MongoDB中，为了操作方便，我们安装了MongoDB可视化工具Robomongo，根据页面所需要展示的数据录入数据库中。由于MongoDB的数据是以BSON结构存储的，类似于我们熟悉的JSON数据格式，在服务端，我们将数据库的数据查询出来展示在一个页面中，即生成一个有JSON格式数据的接口，在前端页面，我们先使用模拟数据去写好样式，用AJAX技术去请求接口中的数据，使用Vue.js将数据渲染到对应的DOM结构中，这样就完成了前后端的连通。

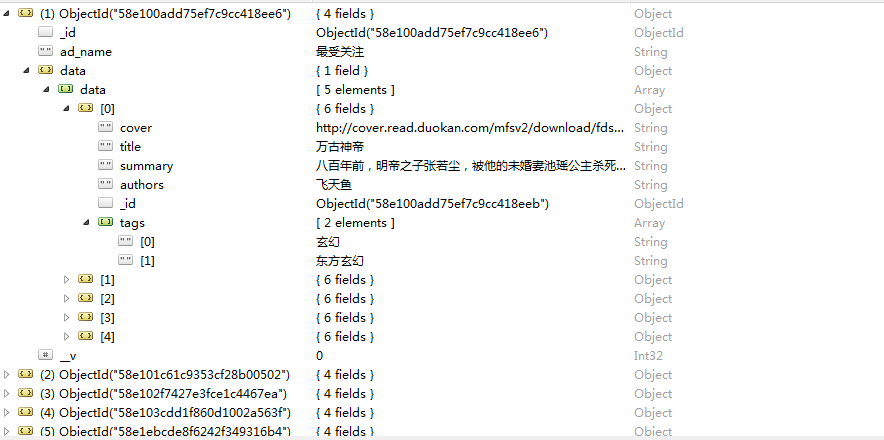


图27. 使用Robomongo录入数据图

男生频道前端与后端核心代码如下：

//将数据库中“男生频道”集合的信息写入接口中

/\*male-start\*/

var maleSchema = new Schema({

ad\_name: String,

data: {

data: [{

cover: String,

title: String,

summary: String,

tags: Array,

authors: String

}]

}

});

var Male = mongoose.model('male', maleSchema);

app.use(controller.get('/ajax/male', function\*(){

this.body = yield Male.find({});

}));

/\*male-end\*/

**代码解释：**

先建立跟数据库中男生频道集合一一对应的模式maleSchema，再建立模型Male，在路由'/ajax/male'中，我们’ this.body = yield Male.find({});’将数据写入页面中。



图28. 男生频道接口中的数据

//使用接口中的数据并使用Vue.js框架

$.get('/ajax/male', function(d){

new Vue({

el: '#app',

data: {

items: d

}

});

},'json');

**代码解释：**

使用Zepto.js中提供的get方法来请求'/ajax/male'中的数据，数据全部保存在参数d中,

Vue方法中的el: '#app'表示在id为app的范围内可以用Vue渲染数据，data表示请求回来的数据，'json'表示数据以json格式解析。

//前端页面使用Vue.js

<div class="new-page channel-page" id="app">

<section class="channel-group main-card" v-for="section in items">

<div class="channel-group\_\_header">

<h2 class="channel-group\_\_title channel-group\_\_title\_left">{{section.ad\_name}}</h2>

</div>

<ul class="book-list">

<li v-for="book in section.data.data">

<div class="u-book">

<div class="cnt">

<div class="book-cover">

<img alt="{{ book.title }}" v-bind:src="book.cover">

<div class="u-tagRT"></div>

</div>

<div class="info">

<h3 class="title single-line">{{ book.title }}</h3>

<p class="summary">{{ book.summary }}</p>

<div class="wrap">

<div class="book-tag" v-for="tag in book.tags" v-if="$index>=0 && $index<3">{{ tag }}</div>

<p class="author single-line">

<span class="author">{{ book.authors }}</span>

</p>

</div>

</div>

</div>

</div>

</li>

</ul>

</section>

</div>

**代码解释：**

‘v-for’代表遍历输出接口中的所有数据，’ <img alt="{{ book.title }}" v-bind:src="book.cover">

’对于数据，我们需要采用Vue.js中规定好的这种形式，一对花括号中包含数据名称，例外的是对于图片地址，我们需要使用v-bind的形式。’ v-if="$index>=0 && $index<3"’表示只输出索引值大于等于0小于3的，这个是为了输出部分数据而设置的，有时候太多数据页面不够美观，所以在本系统中我们使用这种方式，只输出三个标签。

## 5.5 阅读器翻页功能实现

阅读器翻页功能核心代码如下：

//获得上一章内容

var prevChapter = function() {

Chapter\_id = parseInt(Chapter\_id);

if (Chapter\_id == 0) {

return;

}

var cid = Chapter\_id - 1;

gotoChapter(cid);

Util.StorageSetter(Fiction\_id + 'last\_chapter', Chapter\_id);

};

//获得下一章内容

var nextChapter = function() {

Chapter\_id = parseInt(Chapter\_id);

if (Chapter\_id == Chapters.length - 1) {

return;

}

var cid = Chapter\_id + 1;

gotoChapter(cid);

Util.StorageSetter(Fiction\_id + 'last\_chapter', Chapter\_id);

};

**代码解释：**

翻页操作时需要获取当前页的id,并对id进行判断，若id为1，则不允许翻“上一页”，若id为最后一个，则不允许翻”下一页”。

# 系统测试

系统测试是在完成系统的开发后，将系统与外在软件结合，进行功能以及外观上的测试。我们会根据需求分析去对系统进行各方面的测试。在现实工作中，完成一个产品需要经过一系列的流程，产品经理写需求文档，设计师根据需求交互去画设计稿，开发人员根据设计稿开发，开发完成后，系统发布之前最关键的一步就是测试，测试人员为了保证产品到达用户手中时是稳定的，完美的，需要经过一系列的反复测试，在各方面完全确定没有bug后，产品才可以最终发布上线。因此，测试是一个产品上线的必经之路。

## 6.1 测试目的

1. 分析测试结果，得到系统质量评价
2. 研究系统可能存在的缺陷，为开发工程师提供修复和预防的建议
3. 对以后的软件开发避免系统缺陷有参考意义

## 6.2 测试对象

a) 注册功能

b) 管理员添加书籍功能

c) 搜索功能

d) 阅读器字体操作面板功能

## 6.3 测试流程

### 6.3.1测试注册功能

新用户在没有帐号时只能采用游客模式进入书城，因为在想要获取更多权限的时候，我们需要进行注册，界面如下图所示：



图29. 注册界面

根据用户所输入的信息进行合理性校验，手机号码校验提示信息及前后输入密码校验提示图如下所示：

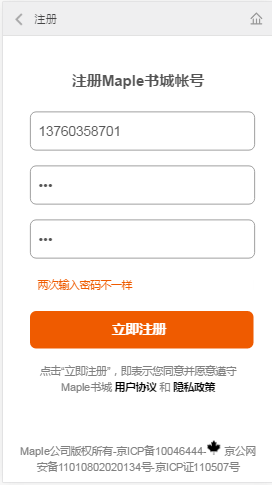
 

图30. 手机号码校验提示信息 图31. 前后输入密码校验提示

### 6.3.2测试管理员添加功能

若为管理员账号登录，即可进入管理员书籍添加页面，如下图所示，当添加成功后，在男生频道页面则可以显示出书籍。

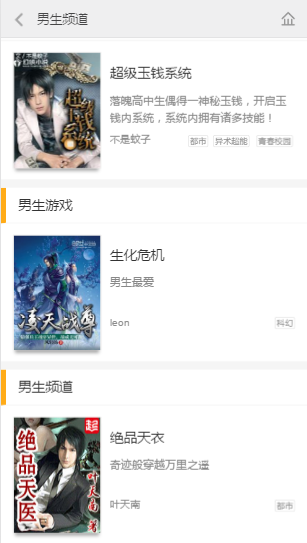
 

图32. 书籍添加页面 图33. 成功展示页

### 6.3.3测试搜索功能

关键字搜索功能，当用户在搜索框中输入关键字时，若数据库中存在，则将书籍封面，书籍名称，作者显示在页面中，若不存在，则显示“没有相应的搜索结果”。

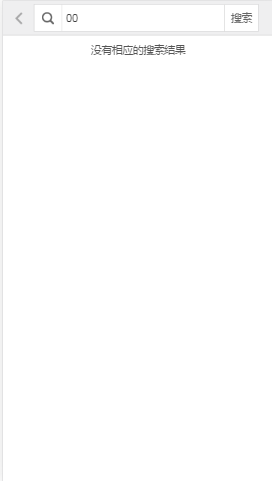
 

图34搜索存在页 图35搜索不存在页

### 6.3.4测试阅读器字体操作面板功能

对字体设置面板中设置字体大小以及背景颜色按钮的记忆功能进行测试，当设置背景颜色后，刷新浏览器，看是否仍然为上一次的设置选项。

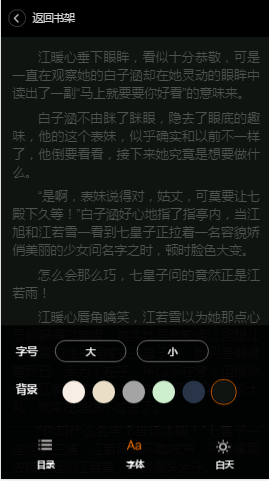
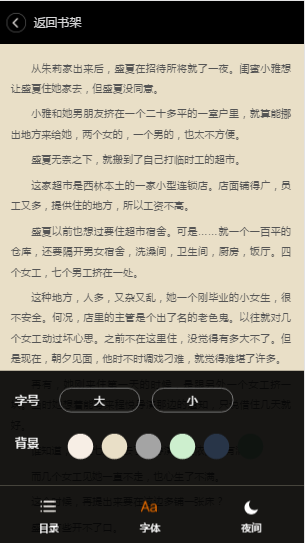
 

图36. 背景图 图37. 字体图

## 6.4 测试结论

经过几轮反复测试，本系统在业务逻辑上以及用户交互，用户体验上没有出现大的问题，基本可以满足用户方便地阅读小说的需求，整个系统采用统一的橘子色调，风格统一，界面协调。当然，在一定程度上，系统没有达到十分完美，在响应速度上，点击翻页按钮稍微有点慢，还需要做到精益求精。总体而言，本系统可以让用户使用。

# 结束语

Web App在当今时代流行，越来越多的网站会开发出Web App版本，因为它开发成本低，更新版本无需经过审核，直接更新。用户直接在浏览器输入URL即可访问，方便快捷。

本文论述了一个系统基于HTML5,CSS3,Zepto.js来开发的Web App电子阅读器系统，描述了整个系统开发的过程，从选题背景，需求分析，系统流程，数据库设计到服务端框架搭建以及系统的具体实现，每一步的实现都经过充分的准备，在开发过程中，遇到的主要难点是服务端接口的开发，由于笔者在学习生活中偏向于前端方面的开发，在本系统中，专注于前端页面样式以及交互便捷性的开发，在服务端方面，采用了Node.js平台去开发接口，开发过程中，对Node.js用法以及服务搭建不太熟悉，通过到网络上查询了大量资源最终解决了难题，在以后的学习中，我会进一步加强对服务端方面的学习，争取取得更大的进步。

经过了一个多月的努力，我独立一个人完成了项目的设计，开发，以及论文的撰写，有过连续调试代码几天没有成功的沮丧，也有过经过多方努力完成一个功能后的喜悦。无论如何，在即将完成论文之时，我的心情是激动的，这一个多月来的开发，让我学到了很多东西，也明白了尝试的重要性。

# 【参考文献】

【1】陈勇. WebApp现状分析及展望[J]. 通信与信息技术, 2012(4):77-78.

【2】于洋. 浅析HTML5的特点及其在移动Web App中的应用[J]. 计算机光盘软件与应用, 2014(24):288-288.

【3】陈梅. 提高用户体验的CSS3技术应用研究[J]. 电脑知识与技术:学术交流, 2015(5):33-34.

【4】[远人](http://www.imooc.com/u/101686/courses?sort=publish). 移动端开发框架Zepto入门[在线]. 慕课网.http://www.imooc.com/learn/229, 2016-05.

【5】彭娜. 基于Node.JS博客系统的设计与实现[D]. 大连理工大学, 2013.

【6】wh1100717. Koa (koajs) -- 基于 Node.js 平台的下一代 web 开发框架[在线]. http://koa.bootcss.com/, 2014-05

【7】虞鹰. 我国成年国民图书阅读率为58.4%[在线]. http://www.huaxia.com/zhwh/whrd/whrdwz/2016/04/4809903.html, 2016-04

【8】Shelley Powers, 夏思雨, 高亮. Node学习指南[M]. 人民邮电出版社, 2014.

【9】riskers. 移动端适配方案(上). https://github.com/riskers/blog/issues/17, 2016-01

【10】Swiper中文网. Swiper使用方法. http://www.swiper.com.cn/usage/index.html, 2015-01

【11】李克. 用于开发管理信息系统的高内聚低耦合通用模块[J]. 电子, 1991(1):68-70.

【12】许阳寅. 渐进增强的前端优化[J]. 程序员, 2010(11):32-32.

# 致谢

在论文即将完结之际，我要在此对那些给予过我无私帮助的人们表示衷心的感谢。首先，，我要感谢我的指导老师秦高德老师，感谢老师在毕业设计开发和毕业论文撰写的整个过程中，对我的积极指导以及关心，能够得到和如此尽职的老师的指导机会，给我的大学生涯画上了一个圆满的句号。在此祝他身体健康，工作顺利。

此外，我要感谢在毕业设计开发过程中对我提供帮忙的师兄师姐以及公司同事，证书他们以过来人的身份给我传授了经验，在我遇到不会的问题时积极为我解答，使得我可以按时完成项目的开发以及论文的撰写，谢谢！

最后，还有感谢互联网时代知识共享给我带来的极大便利，在项目开发期间，我在慕课网，Node开源社区，CSDN博客网站等获得了很多项目开发知识，希望线上学习资源越来越丰富，祝福学习网站越来越好。

**Mobile web eReader**

**【Abstract】**

In the era of Mobile Internet, mobile devices play an increasingly important role in our daily life, and more and more systems will be developed in the form of Web App, because its lightweight and convenient features are popular.

This paper is a mobile web e-reader system, the system is based on HTML5, CSS3, Zepto.js to develop the Web App, the development process is divided into four parts, front-end reader Interface development, front-end bookstore interface development, administrator interface development and service interface development. Users can register to log in, read the books, the administrator can add book details, delete books. The system server uses the Node.js framework --Koa for quick setup.

The system is designed for the use of mobile phone users to read books, users can easily use the phone to read anytime, anywhere.

**【Keyword】**

Mobile web; e-reader; HTML5; CSS3; Zepto.js; Koa; MongoDB

指导老师：秦高德