**中图分类号：TP399**

**论文编号：10006GS0906131**



专业硕士学位论文

**基于Service Worker**

**离线阅读系统设计与实现**

作者姓名 曾梦思

学科专业 计算机技术

指导教师 胡春明 教授

??? 高工

培养院系 计算机学院

**Design and Implementation of Offline Reading System based on Service Worker**

A Dissertation Submitted for the Degree of Master

**Candidate：Zeng Mengsi**

**Supervisor：Prof. Hu Chunming**

**Senior Engineer. Xxx**

School of Computer Science & Engineering

Beihang University, Beijing, China

**中图分类号：TP399**

**论文编号：10006GS0906131**

硕 士 学 位 论 文

基于Service Worker离线阅读系统设计与实现

作者姓名 曾梦思 申请学位级别 硕士学位

指导教师姓名 胡春明 职 称 教授

学科专业 计算机技术 研究方向 计算机技术

学习时间自 2015年03月05日 起至 2017年09月18日止

论文提交日期 2017年09月18日 论文答辩日期 2017年09月28日

学位授予单位 北京航空航天大学 学位授予日期 年 月 日

关于学位论文的独创性声明

本人郑重声明：所呈交的论文是本人在指导教师指导下独立进行研究工作所取得的成果，论文中有关资料和数据是实事求是的。尽我所知，除文中已经加以标注和致谢外，本论文不包含其他人已经发表或撰写的研究成果，也不包含本人或他人为获得北京航空航天大学或其它教育机构的学位或学历证书而使用过的材料。与我一同工作的同志对研究所做的任何贡献均已在论文中作出了明确的说明。

若有不实之处，本人愿意承担相关法律责任。

学位论文作者签名：        日期： 年 月 日

学位论文使用授权书

本人完全同意北京航空航天大学有权使用本学位论文（包括但不限于其印刷版和电子版），使用方式包括但不限于：保留学位论文，按规定向国家有关部门（机构）送交学位论文，以学术交流为目的赠送和交换学位论文，允许学位论文被查阅、借阅和复印，将学位论文的全部或部分内容编入有关数据库进行检索，采用影印、缩印或其他复制手段保存学位论文。

保密学位论文在解密后的使用授权同上。

学位论文作者签名： 日期： 年 月 日

指导教师签名： 日期： 年 月 日

##### 摘 要

伴随着信息化和网络的发展，阅读成为了人们网络生活中不可或缺的一部分。一方面，随着电子阅读的迅速发展，读者对阅读体验要求越来越高。另一方面由于网络技术的发展，HTML5的出现，WEB 应用功能不断丰富和强化，在浏览器中实现原生应用般的体验已不再是纸上谈兵。如何在没有网络，或者网络信号较差的情况下，让读者感觉阅读过程仍然很流畅，如何提高读者的阅读体验，抓住读者群体，这对移动互联网、移动媒体而言是一个巨大的挑战，是推动数字出版物上升到更高的层次的关键，更是一个亟待解决的问题。

本文主要通过对JavaScript的深入研究，利用Web Worker的优势，运用Service Worker技术，在浏览器实现中间层，模拟服务器端操作，拦截用户请求。实现了在浏览器中离线阅读，增强了用户的阅读体验，解决了目前，Web离线应用体验差的实际问题。主要工作如下：

1. 介绍和了选题的背景与意义，分析了国内个研究现状及发展动态。
2. 对电子出版技术进行了概述，对离线Web开发技术进行了分析。

(3) 对Service Worker离线开发技术进行了深入的研究。运用Service Worker，结合本地缓存技术，完成离线应用框架搭建，包括对离线数据的读写和同步操作。

(4) 对离线阅读系统的设计实现进行了详细的描述，包括系统的需求分析、概要设计、详细设计、编码实现、系统测试。

关键词：Web应用，Service Worker，离线阅读，JavaScript

##### Abstract

With the development of information and network, reading has become an integral part of people's network life. On the one hand, with the rapid development of electronic reading, readers have higher and higher requirements for reading experience. On the other hand, due to the development of network technology, the emergence of HTML5, WEB applications continue to enrich and strengthen, and in the browser to achieve a native application like experience is no longer an armchair strategist. How in the absence of network, or poor network signal case, let the readers feel the reading process is very smooth, how to improve the reader's reading experience, grasp the readers, this is a huge challenge to the mobile Internet, mobile media, is to promote the digital publication up to key higher level, more is a problem to be solved.

This article is mainly through the in-depth study of JavaScript, using the advantages of Web Worker, using Service Worker technology to implement the middle layer in the browser, simulate server-side operations, and intercept user requests. It realizes offline reading in browser, enhances user's reading experience, and solves the actual problem of poor Web offline application experience. The main work is as follows:

(1) introduce and select the background and significance of the topic, and analyze the current research situation and development trend in china.

(2) the electronic publishing technology is outlined, and the off-line Web development technology is analyzed.

(3) the offline development technology of Service Worker is deeply studied. Using Service Worker and local cache technology, the offline framework is built, including reading writing and synchronous off-line data.

(4) the design and implementation of the offline reading system are described in detail, including system requirements analysis, outline design, detailed design, coding implementation, and system testing.

**Key words**: Web Applications, Service Worker, Offline Reading, JavaScript

**目 录**

[第一章 绪论 1](#_Toc326079851)

[1.1 论文选题的背景与意义 1](#_Toc326079852)

[1.2 国内外研究现状及发展动态 2](#_Toc326079853)

[1.3 论文的来源、研究目标和内容 5](#_Toc326079854)

[1.4 全文的组织结构 6](#_Toc326079855)

[第二章 电子出版物及离线web应用开发技术 9](#_Toc326079856)

[2.1 电子出版物概述 9](#_Toc326079857)

[3.2.1 技术发展 23](#_Toc326079870)

[3.2.2 主要格式 24](#_Toc326079871)

[2.2 离线WEB应用开发技术 10](#_Toc326079858)

[3.2.1 浏览器缓存 23](#_Toc326079870)

[3.2.2 HTML5本地缓存 24](#_Toc326079871)

[3.2.3 SERVICE WORKERS的工作原理 28](#_Toc326079873)

[3.2.3 本章小结 28](#_Toc326079873)

[第三章 基于Service Workers离线阅读系统总体设计 57](#_Toc326079893)

[5.1 基于Service Workers的离线框架设计 57](#_Toc326079894)

[5.2 基于Service Workers的离线表单提交机制 59](#_Toc326079895)

[5.3 测试 59](#_Toc326079895)

[5.4 本章小结 65](#_Toc326079899)

[第四章 在线期刊发布阅读系统总体设计 36](#_Toc326079880)

[4.1 总体设计 36](#_Toc326079881)

[4.2 网络拓扑结构及安全方案 39](#_Toc326079882)

[4.3 功能模块设计 46](#_Toc326079886)

[4.4 数据库设计 50](#_Toc326079889)

[4.5 本章小结 55](#_Toc326079892)

[第七章 系统的测试 66](#_Toc326079900)

[6.1 测试概述 66](#_Toc326079901)

[6.2 测试目标 66](#_Toc326079902)

[6.3测试方法 67](#_Toc326079903)

[6.3系统功能测试 67](#_Toc326079903)

[6.3系统性能测试 67](#_Toc326079903)

[6.4 本章小结 74](#_Toc326079909)

[第八章 结论与展望 75](#_Toc326079910)

[7.1 论文总结 75](#_Toc326079911)

[7.2 展望 76](#_Toc326079912)

[参考文献 77](#_Toc326079913)

[致 谢 80](#_Toc326079914)

**图 目 录**

[**图1 XXXXX** 3](#_Toc325187973)

**表 目 录**

[**表1 XXXXX** 18](#_Toc326148645)

第一章 绪论

## 1.1论文选题的背景与意义

阅读是人类对知识的获取的一种最重要的途径。对于各种文字信息与知识而言，它们的呈现首先依赖于某些物质载体，而这些物质载体却随着社会生产力与科学技术的发展在不断地演化。从甲骨、木头、石头、青铜器， 到竹简、绢帛、纸张。进入了二十一世纪以来，随着现代科学技术的发展，各种电子屏幕成为了一种新的阅读载体。通过各种电子阅读工具，在各种电子屏幕上进行阅读已发展成为一种新的重要的阅读方式，我们称之为电子阅读，也叫数字阅读。

电子阅读正在迅速的发展，这主要体现在以下几个方面：一、电子阅读工具，如笔记本电脑、智能手机、平板电脑、电子纸书，产量与质量在不断提高，功能不断完善。二、移动互联网技术、电信网技术的快速推进与发展，使得人们可以通过手持阅读设备上网查阅各种信息资料。三、电子出版物的数量也在快速增长，越来越多的阅读内容由传统的纸质载体转移到了各种电子阅读载体上。据市场预测，到2020年，将有90%的图书以电子书和纸质书并存的形式发行。与此同时，进行电子阅读的人群数量也在快速增长。[1]从2005年以来，WEB2.0的草根、互动、参与、分享等理念的提出与移动计算、云计算、阅读终端等技术的相互融合使这上阶段数字出版的发展充满了惊喜，充满了闪光点，无论是在业界的实践，背后的理念支撑，技术的支持都大大超过以前的任何一个阶段，网络的概念不再局限于互联网，终端设备也不再局限于笨重的笔记本台式机，而且更为重要的是草根力量在数字内容的生产上占据重要地位，传统出版面临的压力空前，传统出版在内容的获取和内容的发行的产业链两头都受到了电子出版强有力的挑战。[2]

电子阅读在整个图书市场中的份额持续扩大，越来越多的人逐渐习惯于使用电子设备进行阅读，但是电子阅读的发展还局限在过去，在线电子阅读，局限于有网络的情况下，例如，在火车上，网络信号差的地方，就不能进行阅读。离线电子阅读，就必须提前下载缓存下来，完全没法进行数据的交互和反馈。如何在没有网络，或者网络信号较差的情况下，让读者感觉阅读过程仍然很流畅，如何提高读者的阅读体验，抓住读者群体，这对移动互联网、移动媒体而言是一个巨大的挑战，是推动数字出版物上升到更高的层次的关键，更是一个亟待解决的问题。

随着互联网技术的发展，新一代互联网革命的到来。最新一代HTML标准版本应用，HTML5的普及，使网页的表现性能大为提升，Web应用功能不断丰富和强化，实现了JavaScript技术、CSS技术 与HTML技术的有效整合，给用户更为真切和舒适体验，如离线时也能阅读文章和提交表单数据。基于Web技术的电子数字出版物的在用户交互上，具有很大的优势，给用户带来了新的体验。在线情况下很容易实现，那如何确保离线情况下的用户体验呢？传统的方向，要么是基于应用程序实现，用户必须下载安装相关的应用APP，用户提前把该电子文档下载或者缓存起来，再在离线的环境下进行阅读。这种方式让用户安装APP应用，增加用户的学习成本，占用手机空间。实现用户交互非常困难，给用户带来的离线体验都比较有限，总体来说，都存在一定的缺陷。随着HTML5中新技术的出现，Service Worker对离线请求的支持，以及浏览器产业的发展。而且万维网联盟(World Wide Web Consortium)，W3C发布了PWP的草案，可能为基于Web的数字出版带来新的机会。[3]

随着Service Worker技术的提出，新型的离线阅读使其在Web上实现原生应用成为了可能。它给本地阅读带来了，简便、快速、离线的新体验，Service Worker可以将数据缓存在本地，离线实现数据交互，减少Web应用对网络的依赖。

本文的研究目标：基于Service Worker技术，结合HTML5中本地存储的关键技术，设计与实现支持离线数据交互的Web应用，并在新一代“期刊发布阅读系统”中实现离线阅读功能，验证Service Worker离线技术的有效性。

## 1.2国内外研究现状及发展动态

2015 年是“互联网+”元年，在“互联网+”成为国家战略的背景下，各行各业都在探索如何通过“互联网 +”推动产业转型升级，出版业也不例外。对于当今的出版业来说，互联网已全面渗透到出版产业链的各环节，不再是单纯的技术工具，而是变革发展模式和推进全面升级转型的结构性力量[1]。数字出版是未来出版业的大势，这已经成为学界和业界的共识，但当前我国数字出版业以技术运营商、渠道商、数据库服务商等为主导，传统出版社向数字出版转型之路仍然步履蹒跚[2]。原因有很多，受多方面的限制，下面分析一下。

传统的数字出版产业链由著作权人、内容提供商、技术提供商、服务提供商同、平台运营商、网络运营商、硬件生产商 、渠道商、用户等环节组成。数字内容提供商为产业链上游，服务提供商与平台运营商为产业链中的中游部分，数字内容销售商为产业链下游。上游包括传统出版单位、手机内容提供商和服务商，原创文学网站包括红袖添香、榕树下 、起点中文等。中游包括数字内容出版商、数字技术支持商和网络服务运营商，该部分提供数字出版物的网络技术支撑，存储和网络中的传播。数字内容销售商处于产业链下游，由于靠近消费者，具有十分重要的渠道作用。包括当当、卓越、淘宝国内最大的网上图书电子商务交易平台。

随着网络的普及，传统出版向数字出版的转型，对原有传统产业链形态的冲击，新的产业链形态呈现在我们面前，作者-技术平台-读者。作者直接通过平台将自己的作品思想传播给读者，去掉中间繁琐的环节。

为了促进行业的发展，国际数位出版论坛（IDPF）2007年9月发布EPub正式标准，以取代旧的开放Open eBook电子书标准。EPub（Electronic Publication的缩写，意为：电子出版），是一个自由的开放标准，属于一种可以“自动重新编排”的内容；也就是文字内容可以根据阅读设备的特性，以最适于阅读的方式显示。EPub档案内部使用了XHTML或DTBook （一种由DAISY Consortium提出的XML标准）来展现文字、并以zip压缩格式来包裹档案内容。EPub格式中包含了数位版权管理（DRM）相关功能可供选用。现在各平台已有很多软件支持阅读EPub格式的文档。

Web领域的国际标准化组织，万维网联盟（World Wide Web Consortium， W3C）也做了相关的工作，组织了W3C的数字出版活动，目的是给网站的数字出版物提供产业平台，为开放的Web平台开发商和出版业之间建立必要的桥梁。2015年11月26日，新的便携式Web出版物PWP（Portable Web Publications）草案的发布。目的是为便携式网络出版物在开放式的网络上定义一套数字出版的标准。拉近便携式的离线阅读和Web在线阅读之间的距离。同一个数据出版物有两个动态的表现形式，第一是，可以便携式的离线阅读；第二是，内容可以很容易的同用户在线保存。一个便携式的数字出版物，可以放在网上进行离线的阅读，而不需去重构该出版物的内容。出版商可以选择利用这些发布模式一个或两个，用户可以选择这些消费模式的一方或两方。在线和离线模式之间的无缝连接的包括以下基本功能，例如：交叉引用，用户注释，联机数据库的访问，以及授权和权限管理。

虽然该文档，只是草案，但是这标志着万维网联盟，已经在致力于数字出版物的工作，且带领互联网行业形成统一的标准。

## 1.3论文的主要研究内容

本论文基于Service Worker技术，实现离线应用框架，并运用在“期刊发布阅读系统”中。核心思想是用Service Worker拦截用户请求，将返回的数据存在本地缓存，在浏览器中实现离线数据交互。期刊发布阅读系统包括期刊发布子系统和离线阅读子系统。用户可以在系统中完成一体化操作，不用关注电子刊物的格式和其在网络上的传播方式，只需完成内容编辑和简单的排版制作。刊物编辑完成后，系统将自动生成可用的链接地址和文件入库。读者可打开链接进行阅读。主要研究内容包括：

1. 离线应用框架总体设计。

离线应用框架中包含的主要模块。各模块需要完成的主要功能。

1. 离线数据交互机制。

离线时具体怎么实现数据的查询、新增和修改（同步）。

1. 基于PWP框架的参考实现及相关测试。

参考开放式的网络定义的一套数字出版标准，进行系统的测试工作。

1. 期刊发布阅读系统的设计与实现。

包括期刊发布和离线阅读两个子系统的功能分析、详细设计和编码实现。

## 1.4全文的组织结构

本论文围绕电子出版物，分析现状，跟踪发展动态，应用最新的Web开发技术，实现离线阅读体验，在“期刊发布阅读系统”中进行测试验证。

第二章：电子出版物及离线Web应用开发技术。此章对电子出版物的发展及现状进行了研究，对目前离线Web应用开发技术进行了对比。

第三章：基于Service Worker离线阅读系统总体设计。此章。。。。略。。。。。

第四章：期刊发布阅读系统设计与实现。针对。。。。略。。。。。

第五章：结论与展望。本章概括全文，对本论文的研究成果及完成的工作进行总结，分析目前存在的问题，并对进一步的工作方向进行了展望。

第二章 电子出版物及离线Web应用开发技术

## 2.1 电子出版物概述

几个世纪以来，“书”被定义成了多种形式：期刊，杂志，小册子，报纸，地图册，漫画，各种专辑。我们可以称这些不同名称和表现形式的书为“出版物”：制作成有意义的媒体版本并发行。

目前，国内数字出版行业还没有通用的标准和格式。国内正规的数字化图书馆或电子书厂商大都采用自己专用的电子书文件格式，且他们彼此间通常互不兼容，方正的CEB、书生的SEP、超星的PDG、知网的CAJ、Adobe的PDF等等，且这些格式大多数要下载专业的APP程序，才可以提供电子书的阅读支持。虽然下载了专业的阅读软件，但是在手持阅读器、手机等平台上阅读也面临着格式众多且不统一的问题。

“对用户来讲，买每一家的产品都要下载不同的格式，或购买不同的阅读器，用户使用不方便，且成本也在增加，由于每个阅读器、每款软件都包含厂家的研发成本在里面，而这些成本最终也转嫁到了消费者的头上。同时，对各个厂家的发展也是阻碍，相当于每家只做自己的蛋糕，谁与谁都不能共享，阻碍了整个电子书产业的发展和销售。”中国出版科学研究所出版标准化研究室张书卿指出。电子书格式的标准不一致导致用户必须使用不同的阅读器，这也增加了用户阅读的成本，同时也不利于行业内容的交换和整合。

2015年1月19日“北京开卷信息技术有限公司”发布了《2015年全球图书零售市场报告》（以下简称《报告》），该《报告》称2015年多个国家纸质图书销售实现增长，美国增速2.8%；电子书市场快速发展，电子书销量在英美占比超过20%；而且在电子书市场中，成人虚构电子书转化比例最高。

随着互联网标准的发展，HTML5的出现。HTML5是最新一代HTML的标准应用版本，最突出的特点是网页的表现性能大为提升，WEB 应用功能不断丰富和强化，实现了JavaScript技术、CSS技术 与HTML技术的有效整合，给用户更为真切和舒适体验。

引入离线Web应用API以及网络状态监测可以保障基于此技术的各种Web应用在脱机状态下正常使用。首先需要通过manifest属性制定缓存文件列出基本的资源以供脱机时处于可用状态，以这些资源为基础构建基本的应用缓存，以便需要时及时加载这些资源。其次，引入网络状态监测事件实现基于HTML5的各种应用程序能够在离线与在线间切换自如，借助online属性能够及时准确地判断出网络的在线状态，并构建离线与在线的不同运行模式。

HTML5数据存储支持分析。要想真正实现离线应用，保障Web的部分功能程序运作，必须解决好本地缓存问题，本地数据库和Web Storage是HTML5提供的最为重要的存储方式，能够很好地支持各种数据在浏览器本地的存储与读取。

这些离线的支持也只是本地离线数据的缓存，如果有数据交互的操作，同样是不支持离线操作，这种离线数据缓存不能算是真正意义上的离线应用。

1.只有将应用内容保存到本地才能确保用户在离线状态下也能进行各种数据的填报与编辑，相关的数据和资源包括以下几个方面。首先要有应用界面资源，保证能够在离线状态下应用界面能打开，处理过程中通过缓存清单的方式在用户浏览器本地存储空间里保持，这些重要的资源包括CSS文件、JavaScript文件和HTML5文件，还有重要程序应用界面图片资源。其次，应用数据必须做好及时保存。应用数据保存分为两个类型，一个是原有数据，一个是离线操作数据，在离线的情况下，记录用户操作数据。

2.HTML5和其他技术手段相比具有明显的优势，能够跨越各种平台，带来和在线同样的体验，能够有效改善网络不稳定的问题，增强客户体验，完善web应用程序。加载和操作过程及时完成。用户在访问相关页面时需要加载各种数据，在线时能够及时从服务器中获取并保存到本地存储，确保离线时能够及时从本地读取。

3.实现数据的同步。要做到离线与在线的自行切换，保持数据的一致，才能让用户获得一致的体验，需要借助监听网络状态监测事件来同步操作注册到online上，一旦繁盛 online事件，就会自动实施同步操作。在读取相关数据时， 可以通过读取本地存储中保存的离线记录的形式挖成数据同步，根据记录的数据及时给服务器发出请求，保障操作稳定同步推进。

4.数据存储方式。实现本地数据存储的方式有两种，分别为local storage以及IndexedDB，相对来讲，Local Storage较为容易，二IndexedDB在操作上就相对复杂一些，成本较高，实践起来较为困难，但是和Local Storage相比，IndexedDB有着非常高的查询效率，能够支持数据类型非常丰富，应用领域更为广泛。

5.实现效果分析。通过实践验证，基于HTML5的web离线应用方式能够达到较好的效果，在计算机以及移动设备上多次测试验证，尤其是Chrome以及Firefox效果非常明显，由于IE9不支持离线，效果不好。在移动设备上浏览器一般不支持IndexedDB方式，但是以Local Storage能够达到较好的效果。 总之，依靠HTML5实现离线支持，能够在B/S应用中实现离线存储与在线同步功能，能够有效改善网络不稳定的问题，增强客户体验，完善Web应用程序。在实际操作过程中还需要关注更多的细节，把HTML5具有的诸多功能与用户的个性需求相结合，选择最佳的程序结构和实现方式。

### 2.1.1 技术发展概况

随着读者群逐渐迁移至电子设备，2008年到2010年，两年的时间，电子书销量飙升了1260%。

然而，数字世界的天启永远不会到来，至少不是目前能预测的时间里能到来。分析师曾经预言，到了2015年，电子书将取代打印出版物，但事实却是，数字出版物销量大幅放缓。

在电子书产业中，其产业链由内容提供商（作者、出版社）、内容集成商、通信运营商、硬件和软件终端提供商、第三方支付服务商、读者等多项环节构成。电子书产业平台即以某个产业链环节为主导方，通过硬件或软件终端把产业链中的各个环节和角色有机地连接起来，使电子书的出版、销售、消费得以良性循环发展。目前我国的电子书平台已有不少，如中国移动的“和阅读”、当当网的“当当读书”、百度的“百度阅读”、苹果的“ibooks”,亚马逊中国的Kindle电子书城等。

虽然平台型的电子书产业链整合模式具有不可替代的优势，但目前我国电子书平台在内容资源聚集、商业模式构建、服务品质完善等方面还存在诸多的问题，导致还没有出现，像美国亚马逊这样可以称得上是“生态级”的平台公司，平台建设的道路还很漫长。这也是数字出版物销量大幅放缓的原因之一。

1.优质的内容资源是平台建设的基础

电子书属于典型的内容产业，丰富而优质的产品资源是吸引消费者的关键所在。美国的亚马逊之所以成功，首先就因为就其拥有的海量优质内容资源，并且其中有很大一部分是它自身具有版权的内容。相比之下，我国的电子书平台中，即使是亚马逊中国，虽然号称自己有数十万的电子书资源，但这相比起年出版量就达到20多万种的纸质书来说，覆盖面仍然不算广。而且当前的电子书在质量上良莠不齐：某些作品内容低俗，甚至有些涉及到色情、暴力等内容；编校水平不高，存在诸如大量错别字或字体、字号不统一，行距、字距混乱、乱码等问题，这些都使电子书的阅读体验离纸质书还有相当的差距。

要在优质内容资源聚集上取得优势地位，平台主导方一方面应该以最大的诚意与资金投入争取版权方的支持，而不能将“低价免费”作为获得资源的主要方式，当年汉王科技在投入几个亿，开发硬件新产品的同时，却在内容资源获取上吝于付出，最终被很多版权方，放弃合作，这个教训应该避免；另一方面，平台主导方可以主动出击，在遵守我国现有的出版管理体制下，与出版社进行战略合作，主动参与电子书的出版，完善电子书的出版技术，减轻采购的成本压力，这一点类似于目前各大视频网站大力推行的“自制剧”战略思路。

2.可持续商业模式的构建是平台建设的核心所在

商业平台成功的关键，并不一定是资源有多丰富、投入有多大，而是因为它能通过适合自身的可持续商业模式的构建，兼顾平台各方的利益。如苹果公司的App Store，它在内容资源的获取上放开准入的政策以及“三七分成”（苹果公司三成，应用开发者七成）的利益分配比例调动了应用开发者的积极性，同时将应用与用户账号、硬件相捆绑又确保了应用不容易被盗版侵权，它的云下载功能使用户能够在不同设备之间轻松转移应用产品，平台、产品开发者、消费者的利益都得到了保障。

相比之下，我国有些数字媒体公司通过数年的努力，积累了大量的资源，但因为缺乏可持续的商业模式的构建能力，过着“守着金山过穷日子”的日子，让人叹息。要建立可持续的商业模式，平台主导方应该认真研究互联网商业环境和自身的特点，突出差异化竞争。目前有些领先的电子书产业平台在商业模式探索上已走出了稳健的步伐，比如“百度阅读”在大量推出免费电子书的同时，在电子书内部插入一定数量的广告页，读者在阅读基本不受影响的前提下可以享受高质量的电子书，这种模式比较适合我国互联网消费者比较偏爱免费资源的心理特点；中国移动“和阅读”推出的“包月模式”对于大众化阅读群体来说具有很强的吸引力，用户花几块钱就能看上千本书，积少成多，赢利也非常可观。

3.完善的服务是保证用户数量与黏度的关键

根据3Com公司的创始人，计算机网络先驱罗伯特·梅特卡夫提出的定律：网络的价值等于网络节点数的平方，网络的价值与联网的用户数的平方成正比。这一定律揭示了用户数量对于网站的价值具有决定性的作用。对于电子书平台来说，用户数量同样具有决定性的作用。进一步分析，除了数量，用户对平台的忠诚、活跃程度即黏度也非常关键。

而用户最终选择哪家平台及在哪家平台上最活跃，除了取决于哪家平台具有丰富而优质的内容资源外，关键还是看平台能否提供良好的服务，能否给用户最佳的使用体验。

要建立一个完整且良好的平台型的电子书产业链，从内容资源、商业模式、服务品质等各，我国还有很长的路需要走。

### 2.1.2 主要格式

电子书格式包含很广泛，杂志、TXT文档、PDF、EXE、UMD、HLP、CHM、LIT 、WDL、CEB、ABM 、DOC等等都是电子书，还有HTML的网页也能说成是电子书。

目前，国内数字出版行业还没有通用的标准和格式。国内正规的数字化图书馆或电子书厂商大都采用自己专用的电子书文件格式，且他们彼此间通常互不兼容，方正的CEB、书生的SEP、超星的PDG、知网的CAJ、Adobe的PDF等等，且这些格式大多数要下载专业的APP程序，才可以提供电子书的阅读支持。虽然下载了专业的阅读软件，但是在手持阅读器、手机等平台上阅读也面临着格式众多且不统一的问题。

PDF（Portable Document Format的简称，意思「便携式文件格式」）是由Adobe Systems在1993年用于文件交换所发展出的文件格式。它的优点在于跨平台、能保留文件原有格式（Layout）、开放标准，能自由授权（Royalty-free）自由开发PDF相容软体。PDF主要是为印刷设计，文字图形绝对定位，可制作精美的图文混排。PDF格式很精确，每个人看来都是一样的，但相应地对屏幕大小的适应性就差了，而且PDF中的文字内容也分成一小块一小块，导致在小屏幕上使用reflow技术的效果比较一般。PDF也是在网页上比较常见，对于相对正式一点的电子阅读文件也会选用PDF格式。

为了促进行业的发展，国际数位出版论坛（IDPF）2007年9月发布EPub正式标准，以取代旧的开放Open eBook电子书标准。EPub（Electronic Publication的缩写，意为：电子出版），是一个自由的开放标准，属于一种可以“自动重新编排”的内容；也就是文字内容可以根据阅读设备的特性，以最适于阅读的方式显示。EPub档案内部使用了XHTML或DTBook （一种由DAISY Consortium提出的XML标准）来展现文字、并以zip压缩格式来包裹档案内容。EPub格式中包含了数位版权管理（DRM）相关功能可供选用。

其中包括3种文件格式标准（文件的附档名为.epub），这个格式已取代了先前的Open eBook开放电子书标准。该标准主要是文字为主，关注内容，可根据显示尺寸不同，文字自动流淌重排来适应显示屏，是相对定位。开放格式（zip压缩的一堆html+几个XML文件）以封装一组相关资源至一个单一文件（ZIP）的EPUB容器。EPUB Open Container Format（OCF）3.0，此规范定义了一种档案格式与处理模型，以封装一组相关资源至一个单一文件（ZIP）的EPUB容器。EPUB Media Overlays 3.0，此规范定义了对同步文字和声音的格式与处理模型。

本规范所定义的XHTML文件类型是根据W3C的HTML5规范，除非另行指出，否则这些XHTML文件类型皆继承所有来自HTML 5的语义、结构和处理行为。此外，本规范为W3C HTML5文件类型定义了一组扩展，让作者可以将这些扩展纳入XHTML内容文件之中。

本规范定义了一个简化的处理模式，不需阅读系统支持脚本、HTML 5表单或HTML5 DOM。只要求符合本规范的EPUB阅读系统能够处理符合EPUB内容文件。由于支持脚本与HTML5表单乃是阅读系统的非必要功能，符合的阅读系统可以不必完全于HTML5兼容。

Web领域的国际标准化组织，万维网联盟（World Wide Web Consortium， W3C）也做了相关的工作，组织了W3C的数字出版活动，目的是给网站的数字出版物提供产业平台，为开放的Web平台开发商和出版业之间建立必要的桥梁。2015年11月26日，新的便携式Web出版物PWP（Portable Web Publications）草案的发布[17]。目的是为便携式网络出版物在开放式的网络上定义了一套数字出版的标准。拉近便携式的离线阅读和Web在线阅读之间的距离。同一个数据出版物有两个动态的表现形式，第一是，可以便携式的离线阅读；第二是，内容可以很容易的同用户在线保存。一个便携式的数字出版物，可以放在网上进行离线的阅读，而不需去重构该出版物的内容。出版商可以选择利用这些发布模式一个或两个，用户可以选择这些消费模式的一方或两方。在线和离线模式之间的无缝连接的包括以下基本功能，例如：交叉引用，用户注释，联机数据库的访问，以及授权和权限管理。

## 2.2 离线Web应用开发技术

一个站点或应用的好坏，不仅仅是看漂亮的设计、实用和完善的功能，更是离不开与性能息息相关的加载速度。站点或应用没有做任何缓存机制，页面会因为资源的下载和渲染变得很慢。要提高加载速度，了解相关的缓存机制和充分的利用它就变得必不可少了。Web端的缓存机制其实有多种。服务器端缓存和代理服务器端缓存，采取Web加速技术，减少用户访问的延迟。如采用Web缓存(Web Caching )技术、Web预取(Web Prefetching)技术、CDN ( Content Delivery Network)技术等，以此来降低用户可感知的网络访问延迟，提高网络服务质量。除此之外，还有客户端缓存。下面主要介绍客户端上在浏览器中的缓存。

### 2.2.1 HTTP协议缓存

通过HTTP协议，在客户端和浏览器建立连接时需要消耗时间，而大的响应需要在客户端和服务器之间进行多次往返通信才能获得完整的响应，这拖延了浏览器可以使用和处理内容的时间。这就增加了访问服务器的数据和资源的成本，因此利用浏览器的缓存机制，重用以前获取的数据就变成了性能优化时需要考虑的事情。

为每个资源指定一个明确的缓存策略，用以定义资源是否可以缓存，由谁来缓存，可以缓存多久，并且在缓存时间到期时如何有效地重新验证。当服务器返回一个响应时，它需要在响应头中提供Cache-Control和ETag。

说到浏览器中的缓存机制，其实就相当于HTTP协议定义的缓存机制，因为浏览器为我们实现了它。一般情况下我们会想到HTTP响应头中的Expires，Cache-Control，Last-Modified，If-Modified-Since，ETag这样的与缓存相关的响应头信息。

但是这里我们说服务器返回一个响应时提供必要的Cache-Control和ETag即可。这是为什么呢？

因为Cache-Control与Expires的作用一致，Last-Modified与ETag的作用也相近。但它们有以下区别：现在默认浏览器均默认使用HTTP 1.1，所以Expires和Last-Modified的作用基本可以忽略，具备Cache-Control和ETag即可。

但先不考虑用户的操作的影响，服务器提供Cache-Control和ETag响应头来进行的缓存是如何工作的呢？

使用ETag验证缓存的HTTP响应。服务器通过 ETag HTTP 头传递验证码，大概是像“x123cef“这样的字符串。当浏览器在资源过期后再次请求时，浏览器默认会通过If-None-Match传递ETag的验证码，通过验证码可以进行高效的资源更新检查：如果资源未更改，则不会传输任何数据。ETag就主要用来在响应过期之后，验证资源是否被修改。

服务器在第一次返回响应的时候设置了缓存的时间120s，假设浏览器在这120s经过之后再次请求服务器相同的资源，首先，浏览器会检查本地缓存并找到之前的响应，不幸的是，这个响应现在已经‘过期’，无法在使用。此时，浏览器也可以直接发出新请求，获取新的完整响应，但是这样做效率较低，因为如果资源未被更改过，我们就没有理由再去下载与缓存中已有的完全相同的字节。

于是就到了ETag发挥作用的时候了，通常服务器生成并返回在ETag中的验证码，常常是文件内容的哈希值或者某个其他指纹码。客户端不必了解指纹码是如何生成的，只需要在下一个请求中将其发送给服务器(浏览器默认会添加)：如果指纹码仍然一致，说明资源未被修改，服务器会反悔304 Not Modified，这样我们就可以跳过下载，利用已经缓存了的资源，并且该资源会继续缓存120s。

使用Cache-Control缓存。服务器响应浏览器请求时响应头中的Cache-Control响应头使得每个资源都可以通过Cache-Control。HTTP 头来定义自己的缓存策略，Cache-Control 指令用来告诉我们，那个资源在什么条件下可以缓存，以及可以缓存多久。

一般情况下，浏览器发出的所有 HTTP 请求会首先被路由到浏览器的缓存，以查看是否缓存了可以用于实现请求的有效响应。如果有匹配的响应，会直接从缓存中读取响应，这样就避免了网络延迟以及传输产生的数据成本。然而，如果我们希望更新或废弃已缓存的响应，该怎么办？假设我们已经告诉访问者某个CSS样式表缓存长达24小时(max-age=86400)，但是设计人员刚刚提交了一个更新，我们希望所有用户都能使用。我们该如何通知所有访问者缓存的 CSS 副本已过时，需要更新缓存？

实际上以前没有请求过该资源的新的用户会得到更新的资源，但是请求过资源的用户将在过期时间达到之前一直得到旧的被缓存的资源，直到他手动的去清理了浏览器的缓存。手动清理浏览器缓存这种事可能只有程序员才会做，那么我们要怎么做才能让用户得到更新后的资源呢？其实很简单，我们可以在资源的内容更改后，更改资源的网址，强制用户下载新响应。比如在资源链接后添加参数。

对于缓存机制，现在可以做的有哪些？

我在浏览资料的时候发现了一个caching checklist，比较具有参考价值，我们可以遵循建议合理的利用缓存机制：

1.使用一致的网址：如果在不同的网址上提供相同的内容，那么将会多次获取和存储相同的内容。提示：网址是区分大小写的。2.服务器提供验证码 (ETag)：通过验证码，如果服务器上的资源未被更改，就不必传输相同的字节。3.确定代理缓存可以缓存哪些资源：对所有用户的响应完全相同的资源很适合由 CDN 或其他代理缓存进行缓存。4.确定每个资源的最优缓存周期：不同的资源可能有不同的更新要求。审查并确定每个资源适合的 max-age。5.确定网站的最佳缓存层级：对 HTML 文档组合使用包含内容指纹码的资源网址以及短时间或 no-cache 的生命周期，可以控制客户端获取更新的速度。6.变动最小化：有些资源的更新比其他资源频繁。如果资源的特定部分（例如 JavaScript 函数或一组 CSS 样式）会经常更新，应考虑将其代码作为单独的文件提供。这样，每次获取更新时，剩余内容（例如不会频繁更新的库代码）可以从缓存中获取，确保下载的内容量最少。

### 2.2.2 HTML5本地缓存

用户可离线访问你的应用，这对于无法随时保持联网状态的移动终端用户来说尤其重要。用户访问本地的缓存文件，通常意味着更快的访问速度。仅仅加载被修改过的资源，避免同一资源对服务器多次的请求，大大降低了对服务器的访问压力。

1． App Cache应用程序缓存

1) Web Cache及其局限性

传统的HTML技术通过浏览器网页缓存和Cookie技术在客户端来储存网络服务器的少量数据。但是Cookie只有4KB的存储空间，只能存储用户ID等标志性信息，完全无法满足图书信息的离线存储。Flex等Web前端开发技术，其本地化存储空间可以无限扩展。但是Flash的垃圾处理机制的不完善，使得其在移动终端上的功耗巨大，应用范围无法扩展。新兴的HTML5技术加入了本地存储和离线应用这两个功能，使常规的浏览器只能在线才能正常使用，改变为可以存储数据也可以离线浏览。本地化存储包括Local Storage和IndexedDB等本地静态数据存储。离线存储应用程序缓存，用户只要在缓存清单“manifest”中指明要存储的网络资源，浏览器能够根据这些清单将资源自动缓存在本地，当用户处于在线状态时，浏览器就会对缓存清单中的资源进行本地存储。当用户处于离线状态时，浏览器可以通过Application Cache接口来控制离线缓存，存储过的资源可以继续显示和使用。

如果服务器对离线的资源进行了更新，那么必须更新manifest文件之后这些资源才能被浏览器重新下载，如果只是更新了资源而没有更新manifest文件的话，浏览器并不会重新下载资源，也就是说还是使用原来离线存储的资源。

浏览器在下载manifest文件中的资源的时候，它会一次性下载所有资源，如果某个资源由于某种原因下载失败，那么这次的所有更新就算是失败的，浏览器还是会使用原来的资源。而且浏览器对缓存数据的容量限制可能不太一样（某些浏览器设置的限制是每个站点 5MB）。引用manifest的html必须与manifest文件同源，在同一个域下。并且对于需要计算处理的用户请求或者在线的XHR请求，本地离线缓存技术对此完全是束手无策。[6]

Application Cache（简称App Cache）是为支持Web App离线使用而开发的缓存机制。它的缓存机制类似于浏览器的缓存（Cache-Control和Last-Modified）机制，都是以文件为单位进行缓存，而且文件有一定更新机制。但App Cache是对浏览器缓存机制的补充，不是替代。

使用Application Cache需要配置manifest文件。manifest文件包括，指明缓存入口的CACHE，指明哪些资源必须在线访问的NETWORK，如果index.php无法访问则用404.html代替的FALLBACK。这3大部分组成。

这个过程中有几个问题需要注意：

如果服务器对离线的资源进行了更新，那么必须更新manifest文件之后这些资源才能被浏览器重新下载，如果只是更新了资源而没有更新manifest文件的话，浏览器并不会重新下载资源，也就是说还是使用原来离线存储的资源。

对于manifest文件进行缓存的时候需要十分小心，因为可能出现一种情况就是你对manifest文件进行了更新，但是http的缓存规则告诉浏览器本地缓存的manifest文件还没过期，这个情况下浏览器还是使用原来的manifest文件，所以对于manifest文件最好不要设置缓存。

浏览器在下载manifest文件中的资源的时候，它会一次性下载所有资源，如果某个资源由于某种原因下载失败，那么这次的所有更新就算是失败的，浏览器还是会使用原来的资源。

在更新了资源之后，新的资源需要到下次再打开app才会生效，如果需要资源马上就能生效，那么可以使用window.applicationCache.swapCache()方法来使之生效，出现这种现象的原因是浏览器会先使用离线资源加载页面，然后再去检查manifest是否有更新，所以需要到下次打开页面才能生效。

总结下来，有下列三种方法更新缓存文件。用户清空浏览器缓存、manifest 文件被修改、由程序来更新应用缓存。

2． Web Storage

Web Storage分为Session Storage和Local Storage,是本地持久化存储，单独发展成为独立的W3C Web存储规范。本地持久化存储被设计为用来提供一个更大存储量、更安全、更便捷的存储方法，从而可以代替掉将一些不需要让服务器知道的信息存储到Cookies里的这种传统方法。上面一段是对Web Storage存储机制的官方表述。看起来，Web Storage机制类似Cookies，但有一些优势。

Web Storage是通过存储字符串的Key/Value对来提供的，并提供5MB（不同浏览器可能不同，分Host）的存储空间（Cookies才4KB)。另外Web Storage存储的数据在本地，不像 Cookies，每次请求一次页面，Cookies 都会发送给服务器。

Local Storage对象和Session Storage对象使用方法基本相同，它们的区别在于作用的范围不同。Session Storage用来存储与页面相关的数据，它在页面关闭后无法使用。而Local Storage则持久存在，在页面关闭后也可以使用。

Local Storage与Session Storage主要的区别是存储时间和作用域。另外，严格说来local Storage更像是Cookie一类的本地数据存储。但在标准缓存之外，开发人员可以用浏览器的一些功能来实现自定义的客户端“缓存”。

Session Storage 和 Local Storage 是HTML5 Web Storage API 提供的，可以方便的在Web请求之间保存数据。有了本地数据，就可以避免数据在浏览器和服务器间不必要地来回传递。

Session Storage、Local Storage、Cookies都是在浏览器端存储的数据，其中Session Storage的概念很特别，引入了一个“浏览器窗口”的概念。Session Storage是在同源的同窗口（或tab）中，始终存在的数据。也就是说只要这个浏览器窗口没有关闭，即使刷新页面或进入同源另一页面，数据仍然存在。关闭窗口后，Session Storage即被销毁。同时“独立”打开的不同窗口，即使是同一页面，Session Storage对象也是不同的。当浏览器被意外刷新的时候，一些临时数据应当被保存和恢复。Session Storage对象在处理这种情况的时候是最有用的，比如恢复我们在表单中已经填写的数据。

Web Storage带来的好处：减少网络流量：一旦数据保存在本地后，就可以避免再向服务器请求数据，因此减少不必要的数据请求，减少数据在浏览器和服务器间不必要地来回传递。快速显示数据：性能好，从本地读数据比通过网络从服务器获得数据快得多，本地数据可以即时获得。再加上网页本身也可以有缓存，因此整个页面和数据都在本地的话，可以立即显示。临时存储：很多时候数据只需要在用户浏览一组页面期间使用，关闭窗口后数据就可以丢弃了，这种情况使用Session Storage非常方便。

浏览器本地存储与服务器端存储之间的区别其实数据既可以在浏览器本地存储，也可以在服务器端存储。

浏览器端可以保存一些数据，需要的时候直接从本地获取,Session Storage、Local Storage和Cookies都由浏览器存储在本地的数据。

服务器端也可以保存所有用户的所有数据，但需要的时候浏览器要向服务器请求数据。1.服务器端可以保存用户的持久数据，如数据库和云存储将用户的大量数据保存在服务器端。2.服务器端也可以保存用户的临时会话数据。服务器端的Session机制，如JSP的 Session 对象，数据保存在服务器上。实现上，服务器和浏览器之间仅需传递Session Id即可，服务器根据Session Id找到对应用户的Session对象。会话数据仅在一段时间内有效，这个时间就是Server端设置的Session有效期。

服务器端保存所有的用户的数据，所以服务器端的开销较大，而浏览器端保存则把不同用户需要的数据分布保存在用户各自的浏览器中。浏览器一般只用来存储小数据，而服务器可以存储大数据或小数据。服务器存储数据安全一些，浏览器只适合存储一般数据。

Session Storage 、Local Storage 和 Cookies 之间的共同点和区别如下。

共同点：都是保存在浏览器端，且同源的。

区别：Cookie数据始终在同源的http请求中携带（即使不需要），即Cookie在浏览器和服务器间来回传递。而Session Storage和Local Storage不会自动把数据发给服务器，仅在本地保存。Cookie数据还有路径（path）的概念，可以限制Cookie只属于某个路径下。存储大小限制也不同，Cookie数据不能超过4k，同时因为每次http请求都会携带Cookie，所以Cookie只适合保存很小的数据，如会话标识。Session Storage和Local Storage 虽然也有存储大小的限制，但比Cookie大得多，可以达到5M或更大。数据有效期不同，Session Storage：仅在当前浏览器窗口关闭前有效，自然也就不可能持久保持；Local Storage：始终有效，窗口或浏览器关闭也一直保存，因此用作持久数据；Cookie只在设置的Cookie过期时间之前一直有效，即使窗口或浏览器关闭。作用域不同，Session Storage不在不同的浏览器窗口中共享，即使是同一个页面；Local Storage 在所有同源窗口中都是共享的；Cookie也是在所有同源窗口中都是共享的。Web Storage 支持事件通知机制，可以将数据更新的通知发送给监听者。Web Storage 的 API 接口使用更方便。

3． Web Database

上面介绍的Local Storage和Session Storage，这两个是以键值对存储的解决方案，存储少量数据结构很有用，但是对于大量结构化数据就无能为力了，灵活性不够强大。

对于数据库中处理大量结构化数据，HTML5提供了基于SQL的数据库存储机制，用于存储适合数据库的结构化数据。HTML5 的浏览器客户端存储可以使用Web SQL存储引擎，也可以用IndexedDB。

Web SQL Database，它使用 SQL 来操纵客户端数据库的 API，这些 API 是异步的，规范中使用的方言是SQLlite，悲剧正是产生于此，Web SQL Database规范页面有着这样的声明。这个文档曾经在W3C推荐规范上，但规范工作已经停止了。目前已经陷入了一个僵局：目前的所有实现都是基于同一个SQL后端（SQLite），但是我们需要更多的独立实现来完成标准化。也就是说这是一个废弃的标准了，虽然部分浏览器已经实现，但没有后续跟进。根据官方的标准文档，Web SQL Database存储机制不再推荐使用，将来也不再维护，而是推荐使用App Cache和Indexed DB。

现在主流的浏览器都还是支持Web SQL Database存储机制的。Web SQL Database存储机制提供了一组API供Web App创建、存储、查询数据库。Web SQL Database 规范中定义的三个核心方法：openDatabase：这个方法使用现有数据库或新建数据库来创建数据库对象。transaction：这个方法允许我们根据情况控制事务提交或回滚。executeSql：这个方法用于执行SQL 查询。

由于Web SQL Database规范已经被废弃，原因也很清楚，当前的SQL规范采用SQLite的SQL方言，而作为一个标准，这是不可接受的，每个浏览器都有自己的实现，标准不统一。

Web SQL Database实际上已经被废弃，而HTML5的支持的本地存储实际上变成了Web Storage（Local Storage和Session Storage）与IndexedDB。Web Storage使用简单字符串键值对在本地存储数据，方便灵活，但是对于大量结构化数据存储力不从心，IndexedDB是为了能够在客户端存储大量的结构化数据，并且使用索引高效检索的API。

IndexedDB也是一种数据库的存储机制，但不同于已经不再支持的Web SQL Database。IndexedDB不是传统的关系数据库，可归为NoSQL数据库。IndexedDB又类似于Dom Storage的key-value的存储方式，但功能更强大，且存储空间更大。IndexedDB存储数据是key-value的形式。Key是必需，且要唯一；Key可以自己定义，也可由系统自动生成。Value也是必需的，但Value非常灵活，可以是任何类型的对象。一般Value都是通过Key来存取的。IndexedDB提供了一组API，可以进行数据存、取和遍历。这些API都是异步的，操作的结果都是在回调中返回。

在IndexedDB大部分操作并不是我们常用的调用方法，返回结果的模式，而是请求——响应的模式，比如打开数据库的操作“var request= window.indexedDB.open('testDB')”这条指令并不会返回一个DB对象的句柄，我们得到的是一个IDBOpenDBRequest对象，而我们希望得到的DB对象在其result属性中，这条指令请求的响应是一个 IDBDatabase对象，这就是IndexedDB对象，除了result，IDBOpenDBRequest接口定义了几个重要属性onerror: 请求失败的回调函数句柄，onsuccess:请求成功的回调函数句柄，onupgradeneeded:请求数据库版本变化句柄。

所谓异步API是指并不是这条指令执行完毕，我们就可以使用request.result来获取IndexedDB对象了，就像使用AJAX一样，语句执行完,并不代表已经获取到了对象，所以我们一般在其回调函数中处理。

IndexedDB有个非常强大的功能，就是index（索引）。它可对Value对象中任何属性生成索引，然后可以基于索引进行Value对象的快速查询。 要生成索引或支持索引查询数据，需求在首次生成存储对象时，调用接口生成属性的索引。可以同时对对象的多个不同属性创建索引。

索引的一个好处就是可以迅速定位数据，提高搜索速度，在IndexedDB中有两种索引，一种是自增长的INT值，一种是keyPath：自己指定索引列，我们重点来看看keyPath方式的索引使用。

我们可以在创建object store的时候指明索引，使用object store的createIndex创建索引，方法有三个参数索引名称、索引属性字段名、索引属性值是否唯一。

IndexedDB是一种灵活且功能强大的数据存储机制，它集合了Web Storage和Web SQL Database的优点，用于存储大块或复杂结构的数据，提供更大的存储空间，使用起来也比较简单。可以作为Web SQL Database的替代。不太适合静态文件的缓存。以key-value 的方式存取对象，可以是任何类型值或对象，包括二进制。可以对对象任何属性生成索引，方便查询。较大的存储空间，默认推荐250MB（分Host），比Dom Storage的5MB要大得多。通过数据库的事务（tranction）机制进行数据操作，保证数据一致性。异步的 API 调用，避免造成等待而影响体验。

4． Cache Storage

Local Storage 应是家喻户晓的？但本地存储这个家族可远不止它。前面也说过 Session Storage，现在还有个神奇的 Cache Storage。它用来存储 Response 对象的。也就是说用来对 HTTP 响应，做缓存的。虽然 Local Storage也能做，但是它可能更专业。

Cache Storage 在浏览器上的引用名叫 caches 而不是驼峰写法的 Cache Storage，它定义在 Service Worker 的规范中。Cache Storage 是多个 Cache 的集合，而每个 Cache 可以存储多个 Response 对象。

首先，在 caches 上调用 open 方法就可以异步地，得到一个 Cache 对象的引用。在这个对象上我们可以把 Response 对象 put 进去（参数是一个 URL 和一个 Response 对象）、用 match 方法取出（传入一个 URL 取出对应的 Response 对象）。match 方法不仅可以在 Cache 上调用 Cache Storage 上也有 match 方法，比如上面例子就打开了两个 Cache，都写入一个叫 /hehe 的 URL。在写入操作完成之后，到外部的 Cache Storage 上调用 match 方法来匹配 /hehe，结果是随机的（没找到这个规则在哪里定义的）。

虽然上面的例子中只对 Cache 对象 put 了一个数据，而 Cache 对象本身可以存放更多的 URL/Response 对。并且提供了 delete（用户删除）、keys（用于遍历）等方法。但是 Cache 并不像 Local Storage 一样有 clear 方法，如果非要清空一个 Cache，可以直接在 Cache Storage 上把整个 Cache 给 delete 掉再重新 open。

这套 API 和 Service Worker 一家的，它通常被用于 Service Worker 中，整个设计风格也和 Service Worker 一样都基于 Promise的。

综上所述，总结出HTML5本地缓存的优缺点和适用场景。如下表所示：

|  |  |  |
| --- | --- | --- |
| **缓存机制** | **特点** | **适用场景** |
| 浏览器缓存机制 | HTTP协议层 支持 | 静态文件的缓存 |
| Local Storage | 更大存储量、更安全、更便捷的本地存储方法\持久存在，在页面关闭后也可以使用 | 存储一些用户操作的个性化设置的文本类型的信息和数据 |
| Session Storage | 更大存储量、更安全、更便捷的本地存储方法\存储与页面相关的数据，它在页面关闭后无法使用 | 恢复我们在表单中已经填写的数据 |
| Web SQL | 存储、管理复杂结构数据 | 用IndexedDB替代，  不推荐使用 |
| IndexedDB | 存储任何类型数据，使用简单，支持索引 | 结构、关系复杂的数据存储 |
| Application Cache | 方便构建离线App | 离线App，静态文件的缓存 |
| Cache Storage | 存储Response 对象，和Service Worker配合使用 | 离线App，对 HTTP 响应 |

### 2.2.3 Service Workers的工作原理

### 2.2.4 本章小结

1.JavaScript的单线程特点

JavaScript是单线程的，在某一时刻内只能执行特定的一个任务，并且会阻塞其它任务执行。那么对于类似I/O等耗时的任务，就没必要等待他们执行完后才继续后面的操作。在这些任务完成前，JavaScript完全可以往下执行其他操作，当这些耗时的任务完成后则以回调的方式执行相应处理。这些就是JavaScript与生俱来的特性：异步与回调。

2．HTML5中Web Worker

任何事情都是有双面性的，它有优点，同样也会存在缺点，对于不可避免的耗时操作（如：繁重的运算，多重循环）JavaScript运行就会特别慢。为了解决这个问题，HTML5提出了Web Worker，它会在当前JavaScript的执行主线程中利用Worker类新开辟一个额外的线程来加载和运行特定的JavaScript文件，这个新的线程和JavaScript的主线程之间并不会互相影响和阻塞执行，而且在Web Worker中提供了这个新线程和JavaScript主线程之间数据交换的接口：postMessage和onMessage事件。但在HTML5 Web Worker中是不能操作DOM的，任何需要操作DOM的任务都需要委托给JavaScript主线程来执行，所以虽然引入HTML5 Web Worker，但仍然没有改线JavaScript单线程的本质。

3．浏览器的工作原理

JavaScript引擎线程 JavaScript引擎是基于事件驱动单线程执行的，JS引擎一直等待着，任务队列中任务的到来，然后加以处理，浏览器无论什么时候都只有一个JS线程在运行JS程序。

GUI渲染线程 GUI渲染线程负责渲染浏览器界面，当界面需要重绘（Repaint）或由于某种操作引发回流(reflow)时,该线程就会执行。但需要注意GUI渲染线程与JS引擎是互斥的，当JS引擎执行时GUI线程会被挂起，GUI更新会被保存在一个队列中等到JS引擎空闲时立即被执行。

浏览器事件触发线程事件触发线程，当一个事件被触发时该线程会把事件添加到“任务队列”的队尾，等待JS引擎的处理。这些事件可来自JavaScript引擎当前执行的代码块，如setTimeOut、也可来自浏览器内核的其他线程如鼠标点击、AJAX异步请求等，但由于JS是单线程执行的，所有这些事件都得排队等待JS引擎处理。在Chrome浏览器中，为了防止因一个标签页崩溃而影响整个浏览器，其每个标签页，都是一个进程。当然，对于同一域名下的标签页是能够相互通讯的，具体可看浏览器跨标签通讯。在Chrome设计中存在很多的进程，并利用进程间通讯来完成它们之间的同步，因此这也是Chrome快速的法宝之一。对于Ajax的请求也需要特殊线程来执行，当需要发送一个Ajax请求时，浏览器会开辟一个新的线程来执行HTTP的请求，它并不会阻塞JavaScript线程的执行，当HTTP请求状态变更时，相应事件会被作为回调放入到“任务队列”中等待被执行。

4．Service Worker内部实现机制

Service Worker 是一个脚本，浏览器独立于当前网页，将其在后台运行,为实现一些不依赖页面或者用户交互的特性打开了一扇大门。在未来这些特性将包括推送消息,背景后台同步，geofencing（地理围栏定位），但它将推出的第一个首要特性，就是拦截和处理网络请求的能力，包括以编程方式来管理被缓存的响应。

这个 API 会让人兴奋的原因是，它允许你提供离线体验，而且是开发人员完全可控的离线体验。

Service Worker是基于Web Worker开发实现的，所以Web Worker具备的优缺点，Service Worker同时也继承了这些优缺点。比如说，它不能直接访问 DOM 。但是, Service Worker可以通过postMessage 接口与跟其相关的页面进行通信,发送消息,从而让这些页面在有需要的时候去操纵 DOM。比Web Worker更胜一筹的是，Service Worker 是一个可编程的网络代理，允许你去控制如何处理页面的网络请求。Service Worker 在不使用时将被终止，并会在需要的时候重新启动，因此你不能把onfetch 和 onmessage事件来作为全局依赖处理程序。如果你需要持久话一些信息并在重新启动Service Worker后使用他，可以使用 IndexedDBAPI ，Service Worker支持。如下图是Service Worker的生命周期。

在页面发起http请求时，service worker可以通过fetch事件拦截请求，并且给出自己的响应。w3c提供了一个新的fetch API用于取代XMLHttpRequest，与XMLHttpRequest最大不同有两点：

1. fetch()方法返回的是Promise对象，通过then方法进行连续调用，减少嵌套。ES6的Promise在成为标准之后，会越来越方便开发人员。

2. 提供了Request、Response对象，如果做过后端开发，对Request、Response应该比较熟悉。前端要发起请求可以通过URL发起，也可以使用Request对象发起，而且Request可以复用。但是Response用在哪里呢？在service worker出现之前，前端确实不会自己给自己发消息，但是有了service worker，就可以在拦截请求之后根据需要发回自己的响应，对页面而言，这个普通的请求结果并没有区别，这是Response的一处应用。

第三章 基于Service Worker离线阅读系统总体设计

随着。。略。。关键[8-11]。

## 3.1离线应用框架总体设计

本论文着重于离线应用框架总体设计，通过离线支持提高用户的应用体验。在浏览器中，让用户有原生应用般的体验。当用户在火车上或者其它网络信号不稳定场景下，使用手机电子阅读器阅读时，经常会出现无法正常连接或打开网页响应时间较慢的情况，阅读体验效果差。目前，大多数的网站程序通过缓存网站的静态资源提高网站的访问速度。当网络连接断开的情况下，无法响应用户发送的异步请求和数据交互。本离线框架设计目的是，不管用户在何种网络环境下，读取数据或者提交数据不受网络环境影响，均可以正常使用系统。

本系统框架，对于系统运行过程中使用到的静态资源和动态资源全部进行的缓存。当用户首次使用时，系统自动完成数据加载，并调用Service Worker程序把所需的数据都缓存到缓存中。用户再次访问时，浏览器直接从缓存中返回数据。从而实现用户在线和离线的情况下都可在移动终端浏览器中访问网站。离线应用整体框架示意图如图1所示。

图1 离线应用整体示意图

其中包括四大模块：数据缓存、离线数据访问、离线数据提交和离线数据同步。数据缓存模块需在线时进行数据请求，并把返回的数据缓存在浏览器本地。离线数据访问模块，会处理用户离线时的请求，并正常的返回数据。离线数据提交模块，处理离线情况下用户的提交请求，并把数据存到浏览器本地。离线数据同步模块，完成数据刷新同步操作，保持本地数据与服务器数据的一致性。

以上描述系统的整体模块设计，下面介绍框架的分层设计。为了更好的理解该框架结构，浏览器内部实现代码，可划分为视图、请求控制和数据三大层次结构。这样保证了代码的低耦合、高重用和可维护性。

离线应用分层设计图如图2所示。

请求控制是离线应用分层设计中的核心模块。核心模块的实现依赖于Service Worker技术。Service Worker是如何在浏览器中工作的呢？浏览器内核由两部分组成。一是渲染引擎，决定网页内容和格式在浏览器中的显示效果。另一个是JS引擎，对JavaScript进行解释、编译和执行，使网页达到一些动态的效果。在浏览器中，JS引擎是基于事件驱动单线程执行的，任务队列一直等待任务的到来，然后进行处理，浏览器无论什么时候都只有一个JS线程在运行JS程序。然而Web Worker的出现，使浏览器脚本的处理可以是多线程的。它会在当前JavaScript的执行主线程中利用Worker类新开辟一个额外的线程来加载和运行特定的JavaScript文件，这个新的线程和JavaScript的主线程之间并不会互相影响和阻塞执行，而且在Web Worker中提供了这个新线程和JavaScript主线程之间数据交换的接口。

图2框架分层设计图

Service Worker是基于Web Worker实现，可以进行本地缓存，相当于是一个本地代理，可以减少WEB应用对网络的依赖。其在浏览器中的工作流程图如下图3所示。

请求控制是离线应用分层设计中的核心模块。核心模块的实现依赖于Service Worker技术。Service Worker是如何在浏览器中工作的呢？浏览器内核由两部分组成。一是渲染引擎，决定网页内容和格式在浏览器中的显示效果。另一个是JS引擎，对JavaScript进行解释、编译和执行，使网页达到一些动态的效果。在浏览器中，JS引擎是基于事件驱动单线程执行的，任务队列一直等待任务的到来，然后进行处理，浏览器无论什么时候都只有一个JS线程在运行JS程序。然而Web Worker的出现，使浏览器脚本的处理可以是多线程的。它会在当前JavaScript的执行主线程中利用Worker类新开辟一个额外的线程来加载和运行特定的JavaScript文件，这个新的线程和JavaScript的主线程之间并不会互相影响和阻塞执行，而且在Web Worker中提供了这个新线程和JavaScript主线程之间数据交换的接口。Service Worker是基于Web Worker实现，可以进行本地缓存，相当于是一个本地代理，可以减少WEB应用对网络的依赖。

图3 Service Worker的工作流程图

利用Service Worker，另起一个线程，用来监听所有网络请求，将已经请求过的数据放入缓存，在断网的情况下，直接取用缓存里面的资源。为请求过的页面和图片，展示一个默认值。当有网络的时候，再重新从服务器更新。Service Worker最主要的工作是，对前端视图层发起的请求进行控制，并与浏览器中缓存数据交互，返回数据到前端视图层。

## 3.2离线数据交互机制

### 3.2.1离线数据访问

用户打开浏览器阅读，前端页面发送异步请求，Service Worker程序进行数据拦截处理，返回到前端页面。如图4，基于Service Worker实现的离线数据访问图。

图4 离线数据访问

在浏览器中具体如何实现离线数据访问呢？首先需要在页面上的JavaScript脚本代码中对Service Worker进行注册。告诉浏览器Service Worker脚本在哪里。注册一个Service Worker将会使浏览器在后台启动Service Worker安装过程。通常在安装过程中，可以缓存一些静态资源。如果所有文件都被成功缓存，Service Worker将被成功安装，如果任何文件下载或缓存失败，安装将失败，Service Worker也不会被成功启动。这意味着如果安装Service Worker，需要确保这些静态资源已经在缓存中，否则仍将失败。为了减少失败率，可以在页面加载时候，运用应用程序缓存(Application Cache)，用manifest属性，在缓存配置文件中列出需要加载的静态资源文件，提前加载到缓存中。

如果安装成功，将开始激活过程，这时Service Worker真正价值才开始体现。在激活过程之后，Service Worker将会对处于其作用范围之内的所有页面进行控制，首次注册Service Worker的页面将等到下次被加载时才会被控制。一旦一个Service Worker发挥作用，它将处于两种状态之一，为了节约内存而暂停或当一个网络请求被产生或接收到页面发出消息时执行fetch或message事件处理函数。

Service Worker脚本中定义fetch事件，在event.respondWith里，如果有一个命中的URL，就会返回被缓存的值，否则返回一个实时从网络请求fetch的结果，并且把该请求URL和Response返回的数据存入到Cache Stroage中，以便下次使用。Response返回的数据主要是读者阅读的电子书中的文章数据，该数据格式为HTML，是电子书制作人员在后台管理系统中设计发布的文章。HTML格式的数据中也包括IMG图片文件，在对文章数据进行缓存后，Service Worker也要对文章中的IMG图片请求进行缓存，先提取图片中的SRC，在Worker进程中发送图片链接请求，接收成功返回的请求，存储在Cache Stroage中，等待下次页面需要图片时，直接取本地文件。[27,28，29]URL正则匹配关系如下表所示。

|  |  |
| --- | --- |
| 匹配字符串 | 功能说明 |
| /issue | 获取整本书目录 |
| /getPageList | 根据目录获取每篇文章 |
| <img src= | 获取文章中的图片地址 |

### 3.2.2离线数据提交

另一种离线情景，读者在阅读平台上，电子书进行离线阅读时，需要填写书评、问卷调查等电子表单，填写完毕提交过程中，如果出现网络信号不好，导致提交失败，普通平台会将表单数据清空，需要用户重新填写。但在本系统中用户提交表单数据不受到网络影响。

当用户在离线情况下提交表单数据时，系统先把数据保存在本地队列，在线的后通过Service Worker技术处理离线表单，并自动提交。让用户在浏览器中也有原生应用的体验感。

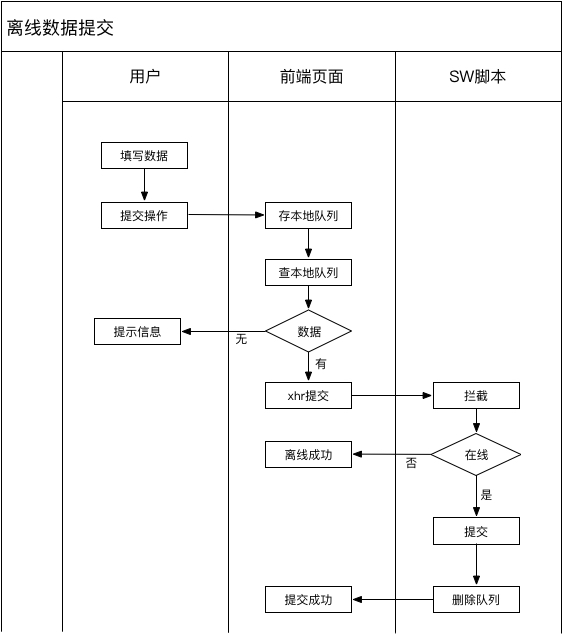
离线数据提交机制需要考虑数据的存储、请求的拦截、用户网络状态的判断和请求的响应四个方面。用户提交表单数据，先把用户数据存储到浏览器缓存IndexDB中，该数据将以本地队列的数据结构形式进行存取操作。IndexDB是一个NoSQL类型的数据库，可在客户端进行大量结构化数据存储，能使用索引高效率搜索数据进行读写和管理操作。而且它的大多操作都是在异步模式下执行，能够让你使用高效的JavaScript事件驱动模块，减少等待，操作更加流畅。

图 5 离线表单提交

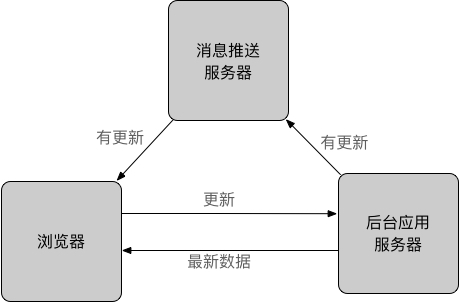
Service Worker脚本程序对用户请求进行拦截，并判断网络状态，若为在线，则提交队列中的数据同时删除已提交的数据；若为离线，则返回给用户网络不可用的提醒，提示用户表单数据已保存在本地缓存。具体流程如图5所示。图5描述了整个数据提交过程在浏览器中的实现流程。

具体的实现如下图6所示。主体文件包括discuss.ejs、discuss.js和service-worker.js。discuss.ejs为前端展现页面，discuss.js前端处理脚本，service-worker.js负责前端xhr请求的处理，并与后台服务器进行数据交互。

图6 离线数据提交时序图

### 3.2.3离线数据同步

离线数据访问和数据提交，分别属于数据的查询和新增。下面介绍数据的修改，也就是离线数据的同步。离线数据的同步，需要做到保持客户端数据与服务器端数据的一致。服务器发现离线数据需要更新，需要先发通知给推送服务器，推送服务器再通知浏览器，浏览器向应用服务器发送更新请求，更新本地离线数据。如图7所示。

图7 离线数据更新示意图

## 3.3测试

。

## 3.4本章小结

本章。。略。。图。

第四章 期刊发布阅读系统设计与实现

传统。。略。。控制技术。

## 4.1 总体设计

在XXXXX系统中，。。略。。性能。

在伺。。略。。定位。

以XXXXX。。略。。H点。

### 4.1.1 总体设计

## 4.2 功能模块设计

XXXXX的缝。。略。。的S型速度规划曲线。

### 4.2.1 功能

## 4.3 数据库设计

### 4.3.1 设计

根。。略。。。。略。。

## 4.4 本章小结

本章。。略。。要求。

第五章 结论与展望

## 7.1 论文总结

嵌入式。。略。。

论文。。略。。主要工作如下：

1．对系。。略。。成本。

2．分析。。略。。的关系。

3．对。。略。。方案。

4．详细。。略。。方法。

5．通过。。略。。任务。

6．对X。。略。。要求。

## 7.2 展望

本文中虽然实现了基于嵌入式的XXXXX控制系统原型机设计，但是该系统还存在着如下一些需要进一步改进的方面：

1．。。略。。。

2．。。略。。

3．。。略。。

4．。。略。。

参考文献

1. 谢誉元，冯炜. “互联网+”对出版行业发展的影响及对策 [J]. 编辑之友， 2015(10).
2. 姚柏年. 数字出版商业模式研究[D]. 华东师范大学， 2012.
3. 邱珊. 使用HTML5WebWorker提高Web的应用性能研究 [J]. 软件导刊，2013(12).
4. 刘耀钦. HTML5中Web Works 应用实践与研究[J]. 郧阳师范高等专科学校学报，2015 (06).
5. 熊紫瑾. 基于HTML5的web离线应用的设计与研究[J]. 信息系统工程，2015(01).
6. 李烨民. 基于HTML5的前端本地化存储技术[J].成都大学学报(自然科学版) ，2012.3(31)
7. 罗大晖，陈娟.基于HTML5 的 Web 离线应用研究与实现[J].计算机应用与软件，2012(12)
8. 严伟中. 关于HTML5的核心技术研究与应用[J]. 网络安全技术与应用， 2014(03).
9. 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).
10. 章斓. 基于HTML5的Web App的开发与探索[J]. 长沙大学学报，2015(05).
11. 刘耀钦. 基于HTML5的Web离线应用研究与探讨[J]. 河南工程学院学报(自然科学版)，2015(01)
12. 刘华星，杨庚.HTML5－下一代Web开发标准研究[J]. 计算机技术与发展，2011，21(08).
13. 邱珊. 使用 HTML5 WebWorkers提高 Web 的应用性能研究[J]．软件导刊.2013(12).
14. (荷) 柳伯斯 (Lubbers，P.)， (美) 阿伯斯 (Albers，B.)， (美) 萨利姆 (Salim，F.)著. HTML5程序设计[M].人民邮电出版社.2016(3): 254-268
15. 崔华俊. 面向移动应用的HTTP内容缓存技术研究与实现[D]. 南京师范大学， 2015.
16. 张宇博， 饶培伦. 基于iOS系统的平板电脑电子阅读软件界面设计要素研究[J]. 人类工效学， 2013， 19(4):42-46.
17. MarcosCaceres，KennethRohdeChristiansen，MounirLamouri，AnssiKostiainen.Web App Manifest[EB/OL]，https://w3c.github.io/manifest/，2013-12-17/2016-12-12
18. 张家重，荆秀丽，许金源. 一种基于Web Worker实现的Web应用中间件异步嵌套调用方法[P].2015.6
19. 于春梅.基于Web Worker的浏览器扩展程序的特权分离研究与实现[D].2014.5
20. 杨舒然. 云南瑞升烟草技术集团有限公司网上阅读系统的设计与实现[D]. 电子科技大学， 2014.
21. 夏翠娟，张燕.图书馆移动阅读服务的新契机:HTML5 和 CSS3[J].现代图书情报.2012(5)
22. Alex Russell，Jungkee Song，Jake Archibald. Service Workers W3C Working Draft 25June2015[EB/OL]，https://www.w3.org/TR/service-workers/，2015-02-05/2015-6-25
23. Deborah Kaplan， Invited Expert.Digital Publishing and Accessibility in W3C Documents[EB/OL].https://www.w3.org/TR/2016/NOTE-dpub-accessibility-20160503/，2016-05-03
24. Markus Gylling，Ben De Meester，Ivan Herman，Tzviya Siegman，Dave Cramer.Web Publications(W3C Editor's Draft 20 December 2016) [EB/OL]. [https://w3c.github.io/dpub-pwp/，2016-12-20](https://w3c.github.io/dpub-pwp/,2016-12-20)
25. Ivan Herman， W3C.Portable Web Publications: Technology Challenges [EB/OL]. https://www.w3.org/2016/Talks/W3CTrack-IH/， 2016-04-13
26. Matt Gaunt. Service Workers: an Introduction[EB/OL]. https://developers.google.com/web/fundamentals/getting-started/primers/service-workers， 2016-12-21
27. Sights.TheHTML5test[EB/OL].http://html5test.com/results/desktop.html， 2013-3-20
28. WhatWG. HTML5 living standard[EB/OL]. http://www.whatwg.org/specs/web-apps/current-work/multipage/，2016.12
29. W3schools．HTML5 教程[EB/OL].<http://www.w3school.com.cn/html5/index.asp，2016.12>
30. Sheppy. Social service worker API reference[EB/OL]. https://developer.mozilla.org/en-US/docs/Mozilla/Projects/Social\_API/Service\_worker\_API\_reference，2016-8-18

致 谢

论文即将完成之时，首先我要感谢我的指导老师白跃彬教授。本论文从选题、方案设计、到最终论文的完成，白教授都给予了我极大的启迪和精心的指导，同时也为我指出论文中的问题，使我能够及时得以改正。白教授严谨的治学态度、渊博的知识、开拓创新的思维方式令我钦佩，使我受益非浅，必将对我今后的工作学习产生深刻的影响。在这里我向白教授表示深深的谢意和良好的祝愿。

同时我还要感谢北京大豪科技股份有限公司胡文海博士，是他把我带进了电机控制领域，他认真的工作作风、忘我的工作精神、丰富的工作经验为我树立了榜样，让我知道如何做一名合格的科学工作者。在两位老师潜移默化的影响和帮助下，我开阔了视野，拓宽了思路，为今后的工作莫定了良好的基础。

。。略。。

其次感谢班主任王瑾玫老师和工程硕士GS09061班的所有同学们，感谢他们营造了一个团结、向上的学术氛围。感谢他们对我学习工作的帮助，使我度过了一段愉快的学生生活。

。。略。。

最后衷心感谢所有关心、爱护、帮助过我的亲人和朋友们。