**中图分类号：TP399**

**论文编号：10006GS1406236**



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

刘炳林 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. Liu Binglin**

School of Computer Science & Engineering

Beihang University, Beijing, China

**中图分类号：TP399**

**论文编号：10006GS1406236**

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

伴随着网络和数据信息化的发展，阅读成为人们网络生活中不可或缺的一部分。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。随着电子阅读的发展，读者对阅读体验要求越来越高。但是电子阅读的发展还局限在过去，如电子书平台和格式众多不统一，而且阅读局限于有网络的情况下。选择什么样的通用的阅读平台，如何在没有网络，或者网络信号较差的情况下，让阅读过程仍然很流畅，如何提高读者的阅读体验，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。通过对网络出版物PWP规范的深入分析，实现整体架构。并在书刊发布阅读系统中运用实现。主要工作如下：

(1) 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修改操作。

(2) 基于便携式网络出版物PWP规范，对网络出版物需求进行分析并实现总体框架设计。

(3) 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)

[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 在线书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 移动应用开发模式架构图 17](#_Toc493600634)

[图 2 新一代Web App架构图 18](#_Toc493600635)

[图3 离线应用整体示意图 18](#_Toc493600636)

[图4框架分层设计图 19](#_Toc493600637)

[图5 Service Worker的工作流程图 20](#_Toc493600638)

[图 6 离线数据访问 22](#_Toc493600639)

[图 7 离线表单提 24](#_Toc493600640)

[图 8 离线数据提交时序图 25](#_Toc493600641)

[图9离线数据更新示意图 26](#_Toc493600642)

[图10 PWP框架设计图 33](#_Toc493600643)

[图 11 系统用例图 35](#_Toc493600644)

[图 12 系统整体架构图 35](#_Toc493600645)

[图 13 系统分层架构图 36](#_Toc493600646)

[图 14 系统功能模块图 38](#_Toc493600647)

[图 15 数据表关系图 39](#_Toc493600648)

[图 16 Postgresql体系结构和进程视图 40](#_Toc493600649)

[图 17 在线设计实现效果图 41](#_Toc493600650)

**表 目 录**

[**表1 缓存机制对比表** 12](#_Toc493618550)

[**表2 移动应用开发模式对比表** 16](#_Toc493618551)

[**表3 正则匹配关系表** 23](#_Toc493618552)

[**表4 浏览器列表** 27](#_Toc493618553)

[**表5 浏览器兼容性测试结果** 27](#_Toc493618554)

[**表6 读者的需求** 30](#_Toc493618555)

[**表7 出版商和作者的需求** 31](#_Toc493618556)

[**表8 打包的WEB出版物** 32](#_Toc493618557)

[**表9 书刊阅读部分测试用例** 42](#_Toc493618558)

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。随着现代科学技术高速的发展，进入21世纪以来，各种电子设备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]自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。

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[2]。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]。

一方面，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。

另一方面，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。

## 1.2 论文的主要研究内容和目标

本论文基于Service Worker技术，实现离线应用框架，参考PWP需求实现设计，并运用在“书刊发布阅读系统”中。离线应用核心思想是用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。书刊发布阅读系统包括书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. 离线阅读应用设计。应用框架总体设计离线数据交互机制。

离线数据交互机制，包括离线时具体怎么实现数据的查询、新增和修改（同步）。离线应用框架总体设计，包含的主要模块，各模块需要完成的主要功能。

1. 基于PWP框架的参考进行分析和设计。

参考开放式的网络上定义的数字出版标准，进行分析，实现网络出版的的整体框架设计。

1. 书刊发布阅读系统的设计与实现。

包括书刊发布阅读系统的需求分析、概要设计、详细设计和编码实现。

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，对PWP参考实现进行分析，并在新一代“书刊发布阅读系统”中实现离线阅读功能，验证Service Worker离线技术的有效性。

## 1.3 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，实现离线阅读应用，并参考PWP进行需求分析和设计，最后在“书刊发布阅读系统”中实现。

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。对三大类移动开发框架进行对比分析，分析原Web APP架构的特点，实现新一代Web APP架构。对离线数据交互机制进行分析，实现离线应用框架的总体设计。

第四章：基于携式网络出版物PWP规范，对网络出版物需求进行分析并实现总体框架设计。

第五章：书刊发布阅读系统设计与实现。需求分析、概要设计、详细设计与实现并参考PWP框架进行离线体验测试。

第六章：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 研究现状及相关技术

## 2.1 数字出版概述

几个世纪以来，“书”被定义成了多种形式：书刊，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。

随着科学技术的发展，网络电子越来越发达，出版形态也发生了巨大的变化。“电子出版”、“网络出版”、“新媒体出版”等概念出现在我们的生活中，数字出版没有统一的定义。下面以桌面出版-电子出版-网络出版-数字出版这样一个过程，分析数字出版的历史沿革经历。

### 2.1.1 技术发展概况

我国在20世纪70年代开始采用计算机进行汉字信息处理，80年代实现版面编辑、数据存储和软件检索等功能，形成了初期的中文电子出版，拉开了数字出版的序幕。这一时期的出版形态为桌面出版，简单的利用计算机等高科技电子技术对书籍报纸杂志等纸质媒体进行编辑出版。随着技术的发展，以磁带为介质的音频出版，变是普遍，同时，多媒体光盘也变得盛行。90年代，在全球多媒体光盘的出版热潮中，以CD-ROM为代表的电子出版物被多媒体公司和电子出版社大量生产。2000年以来，随着网络信息技术的迅速发展，网络传播显示出超越其他媒介的优越性。网络出版的概念也随之产生。根据中国出版科学研究所发布的历年中国数字出版产业年度报告统计，21世纪第一个十年，数字出版规模从15.9亿，上升到1051.79亿，2010年的数字出版总收入约是2002年总收入的66倍，年增加速度为68.88%。然而在这种爆发性的增长之下，我们在量的背后更应该看到一些质的产生。特别是2005年以来，WEB2.0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一阶段数字出版的发展充满了惊喜和闪光点，网络的概念不再局限于互联网，终端的设备也不再局限于笔记本和台式机。21世纪第一个十年，是数字出版物迅速发展的时期，然而，在第二个十年里，数字出版物的销量大幅放缓。从发展趋势看，在相当时期内，数字媒体还不能完全取代传统出版，数字出版与传统出版会共存，数字出版与传统出版呈现出融合共生、互补互动发展的趋势。

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。产业链上游为数字内容提供商，产业链中的中游部分为服务提供商与平台运营商，产业链下游为数字内容销售商。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。

从产业角度分析，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。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。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。

技术角度的数字出版主要侧重于数字出版的软件、阅读器、电纸书、系统浏览器等技术的研发。如何制作数字出版物。目前，在新媒体、新技术的推动下，全球出版业正在经历一场深刻的数字化转型。然而出版单位和新闻单位自主研发的却很少。这主要是由于新闻出版业多年从事的主要是内容生产，传统的编审校已成为习惯性工作。出版单位的人才培养和资金投入都集中到业务方向上，对于技术的研发投入和积累都非常少，仅有的技术人员也多半是负责公司的信息系统建设，对于与主板业相关的高新技术研发严重缺失规划和投入。

从信息传播视角的分析数字出版。涉及未来学、传播学、社会学领域中。计算机可以方便地进行信息的转换、复制和传输，有利于信息资源的自由流通。网络化形成的虚拟空间具有互动、开放、共享、参与的特点，使得数字出版与传统出版相比出现了信息传播的新特征。在之前，读者只是被动地接受信息，而无法自行发布信息。虽然目前有一些阅读软件上实现了，读者与读者之间的交流。但这些功能仅限于阅读软件平台内部的信息交流，没有行成统一的标准。

### 2.1.2 主要格式

电子书格式，是对使用电子书时的文件编码方式，文件结构的一种约定，便于区分。广义的可理解为，可以在电子设备上打开进行阅读的一种格式化文件载体，如TXT、HTML、EXE都可称为电子书格式。

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，且这些格式的电子书阅读，大多数要下载对应的安装程序，才可以阅读。虽然下载不同的阅读软件可以进行阅读，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。

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，2007年9月国际数位出版论坛（IDPF）发布EPub正式标准，以取代旧的电子书标准。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；也就是根据阅读设备的特性，文字内容可以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。现在各平台已有很多软件支持阅读EPub格式的文档。

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HTML和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 3.0，此规范定义了一种档案格式与处理模型，一组相关资源至一个单一文件（ZIP）被封装为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HTML5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、HTML5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。

EPUB格式的电子书很好看，可以重新排版，成为目前比较完善且统一的电子书格式。所有主流的操作系统平台都支持该格式。

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。

## 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，需要先了解相关的缓存机制并充分的利用它。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，包括浏览器缓存和HTML5本地缓存。浏览器缓存也可称为HTTP缓存，属于协议层缓存。HTML5，是新一代的 HTML 标准，加入很多新的特性，离线存储（也可称为缓存机制）是其中一个非常重要的特性。离线本地缓存属于应用层缓存。包括Local Storage、Session Storage、IndexedDB、Web SQL、、Cache Storage和Application Cache。

### 2.2.1 浏览器缓存

浏览器缓存机制是指通过 HTTP 协议头里的 Cache-Control（或 Expires）和 Last-Modified（或 Etag）等字段来控制文件缓存的机制。它是 WEB 中最早的缓存机制了，在 HTTP 协议中定义，通过浏览器来实现。

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Expires，Cache-Control，Last-Modified，If-Modified-Since和ETag。

Cache-Control 用于控制文件在本地缓存有效时长。最常见的，比如服务器回包：Cache-Control:max-age=600 表示文件在本地应该缓存，且有效时长是600秒（从发出请求算起）。在接下来600秒内，如果有请求这个资源，浏览器不会发出 HTTP 请求，而是直接使用本地缓存的文件。

Last-Modified 是标识文件在服务器上的最新更新时间。下次请求时，如果文件缓存过期，浏览器通过 If-Modified-Since 字段带上这个时间，发送给服务器，由服务器比较时间戳来判断文件是否有修改。如果没有修改，服务器返回304告诉浏览器继续使用缓存；如果有修改，则返回200，同时返回最新的文件。Cache-Control 通常与 Last-Modified 一起使用。一个用于控制缓存有效时间，一个在缓存失效后，向服务查询是否有更新。

Cache-Control 还有一个同功能的字段：Expires。Expires 的值一个绝对的时间点，如：Expires: Thu, 10 Nov 2015 08:45:11 GMT，表示在这个时间点之前，缓存都是有效的。

Expires 是 HTTP1.0 标准中的字段，Cache-Control 是 HTTP1.1 标准中新加的字段，功能一样，都是控制缓存的有效时间。当这两个字段同时出现时，Cache-Control 是高优化级的。

Etag 也是和 Last-Modified 一样，对文件进行标识的字段。不同的是，Etag 的取值是一个对文件进行标识的特征字串。在向服务器查询文件是否有更新时，浏览器通过 If-None-Match 字段把特征字串发送给服务器，由服务器和文件最新特征字串进行匹配，来判断文件是否有更新。没有更新回包304，有更新回包200。Etag 和 Last-Modified 可根据需求使用一个或两个同时使用。两个同时使用时，只要满足其中一个条件，就认为文件没有更新。

### 2.2.2 HTML5本地缓存

新一代HTML标准，HTML5，加入了离线存储的新特性。与传统的浏览器缓存比较，该特性并不强制要求用户访问网站。本地缓存，属于应用层缓存，在应用程序代码中实现。

1． App Cache应用程序缓存

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当对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。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。

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。

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。

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。

3． Web SQL Database

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。当前的SQL规范采用SQLite的SQL方言，而作为一个标准，每个浏览器都有自己的实现，标准不统一。Web SQL Database规范已经被废弃，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。

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。

4. Indexed Database

Indexed Database (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。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不是传统的关系数据库，可归为NoSQL数据库。IndexedDB又类似于Web 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。

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。

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。

5． 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：

**表 1 缓存机制对比表**

|  |  |  |
| --- | --- | --- |
| **缓存机制** | **特点** | **适用场景** |
| 浏览器缓存 | HTTP协议层 支持 | 静态文件的缓存 |
| Application Cache | 方便构建离线App | 离线App，静态文件的缓存 |
| Local Storage | 更大存储量、更安全、更便捷；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；存储与页面相关的数据，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 |
| 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

在2014年，W3C公布了Service Worker的草案，Service Worker提供了很多新的能力，使得Web APP拥有与Native APP相同的离线体验、消息推送体验。2015年，Google提出PWA，全称是 Progressive Web App，一个具有响应式布局的Web应用，可以离线工作，并能够安装到设备的主屏幕上。其实是在主屏幕上添加该Web应用的快捷方式。其中的实现离线的核心就是Service Worker，一个Service Worker是一段运行在浏览器后台进程里的脚本，他独立于当前页面，提供了那些不需要与Web页面交互的功能在网页背后悄悄执行的能力。在将来，基于它可以实现消息推送，静静更新以及地理围栏等服务，但是目前它首先要具备的功能是拦截和处理网络请求的功能，包括可编程的消息缓存管理能力，是PWA的核心。

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。

## 2.3 本章小结

本章节主要介绍了在客户端中，实现离线缓存的各种方法。浏览器缓存，在HTTP协议报文头中定义，通过浏览器实现。HTML5本地缓存，Web Storage存储机制、Web SQL Database 存储机制、Application Cache（AppCache）机制、Indexed Database （IndexedDB）。还有应用层实现的可编程的，基于Service Worker的离线缓存。

应用程序缓存为应用带来三个优势：离线浏览，用户可在应用离线时使用它们。速度，已缓存资源加载得更快。减少服务器负载，浏览器将只从服务器下载更新过或更改过的资源。

第三章 离线阅读应用设计

本章首先分析目前三大移动应用开发模式的优缺点。并将Service Worker技术应用Web应用开发中，对离线应用需求进行分析，实现离线应用框架的总体设计,分别对离线数据访问、离线数据提交、离线数据同步进行实现。

## 3.1移动应用框架分析

随着移动互联网的发展，移动应用得到了广泛的应用。形成了三大开发模式：原生应用（Native App）、Web应用（Web App）、混合应用（Hybrid App）。

Native App，以基于智能手机本地操作系统如iOS、Android、黑莓等等并使用原生程式(SDK)编写运行的需要用户安装使用的第三方应用程序；Web APP，以HTML+JS+CSS等Web技术编程，代码运行在移动端浏览器中，通过该移动端浏览器来调用Device API（取决于HTML5未来的支持能力）的不需要用户安装的应用程序；Hybrid App，同时使用网页语言(Web技术)与程序语言(Java、Objective-C等)开发，通过应用商店区分移动操作系统分发，需要用户安装使用的移动应用。三大模式对比如下表2所示。

**表 2 移动应用开发模式对比表**



从上表可以看出，Web App和Native App还存在一定的差距，尤其是对网络的依赖性更高。Service Workers的出现，改变了这一现状。Service Workers提供了很多新的能力，使得Web App拥有与Native App相同的离线体验、消息推送体验，使Web应用能够更像原生应用。现在Service Workers的最佳使用场景是提供离线能力。开发人员可以注册一个Service Workers作为网络代理提供网络拦截。即使没有可用的网络时，这个代理也能够对缓存的数据和资源或者是已经生成的内容做出响应。

图 1 移动应用开发模式架构图

Native App和Hybrid App的优势就在于，可以在客户端中构建数据模型，处理业务逻辑。如图1所示。

在出现Service Workers之前，要开发离线应用，唯一的选择就是开发原生应用，针对不同的移动操作系统开发不同的项目代码。例如，在IOS平台上用Objective-C开发的应用；在Android平台上用Java开发的应用等。同一个应用，不仅需要用多种移动设备进行测试，增加了开发成本高；而且还需要用不同的语言开发，也增加了维护成本。为了解决原生应用开发中的缺陷，出现了Hybrid App。Hybrid App是半原生半Web的混合类App。需要下载安装，看上去类似Native App，但只有很少的UI Web View，访问的内容是 Web。Hybrid App 极力去打造类似于Native App 的体验，但仍受限于技术，网速，等等很多因素。尚不完美。

Service Workers的到来，使在浏览器中开发离线应用成为了可能。也就是可以在客户端浏览器中构建数据模型和处理业务逻辑。新一代Web App架构图如图2所示。



**图 2 新一代Web App架构图**

浏览器可以代替应用客户端，在前端代码中编程实现页面展示，同时也对数据进行操作，而且不依赖于网络。

## 3.2离线应用框架总体设计

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3所示。

图3 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4所示。

图4框架分层设计图

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。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5 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。

## 3.3离线数据交互机制实现

应用程序在没有网络连接的情况下依然能够显示信息，而在连上网络的情况下需要刷新数据。要实现这一点，需要在移动设备上对数据进行一定程度的持久化，并且通常需要保留一段时间。离线数据交互，可理解为对数据的查询、新增和修改。分别对应离线数据访问、提交和同步。离线数据访问涉及阅读数据，离线数据提交涉及表单提交数据，离线数据同步，涉及阅读数据和表单数据。

### 3.3.1离线数据访问

离线数据访问可理解为在离线情况下对数据请求，需要在设备上进行数据的持久化也就是数据的存储。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，这是离线数据访问的基本实现过程。详细如图6，基于Service Worker实现的离线数据访问图。

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。

**图 6 离线数据访问**

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正则匹配关系如表3所示。

**表3正则匹配关系表**

|  |  |
| --- | --- |
| 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 |

### 3.3.2离线数据提交

离线情况下进行数据的提交，需要客户端能够暂存数据，在线时将数据提交到后台服务器。离线情景如下，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。

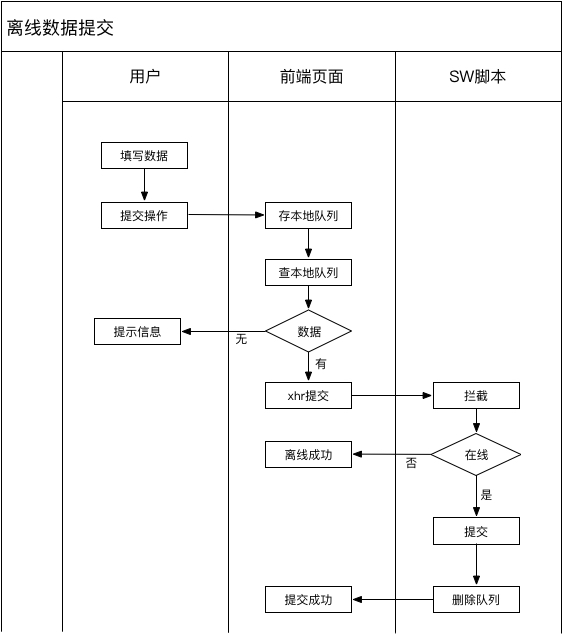


图 7 离线表单提交

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7所示。图7描述了整个数据提交过程在浏览器中的实现流程。

具体的实现如图8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 8 离线数据提交时序图

### 3.3.3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。离线数据的修改，属于离线数据的同步，同时也是数据操作中最为复杂的一部分。

对缓存中的数据进行同步有三种不同的“策略”：

1.网络优先

总是尝试从服务器上获取数据，如果无法从服务器上获取，再转而从本地缓存中获取数据。如果希望总是显示最近的、经过更新的信息，那么可以选择这种策略。

2.本地优先

完全不会从网络上获取数据，而是通过本地缓存进行获取。如果应用能够接受显示被缓存的数据而不存在什么风险，那么这种方式非常适合。另一方面，这种方式的用户体验更好，因为通常来说不会产生任何延迟。

对于性能与用户体验的改进使得本地缓存在许多场景中非常实用，这种实用性的关键条件在于数据无需进行实时显示。能够将数据在本地缓存中保留的时间越长，这种方式的优点就越突出。

3.混合 / 智能

这种方式在从服务端获取数据之前先从本地缓存中返回结果。可以选择等待服务端的通知，或是在后台对服务进行轮询的方式，对本地的缓存数据进行刷新。 这种机制能够在性能与用户体验之间取得一种良好的平衡，而它仍然会对本地缓存进行定期刷新，避免了为用户显示“过期”数据的风险。

此外，通过某种服务端缓存的支持，可以有效地弥补本地缓存的不足之处。正如HTTP缓存一样，当需要从服务器获取数据的时候，客户端可以通过发送一个“修订号”以确认该数据是否已经被更新了。服务端将检查客户端所发送的修订号是否与服务器上的数据一致，根据结果不同，或者通知客户端不必更新数据，或者将最新的数据返回。

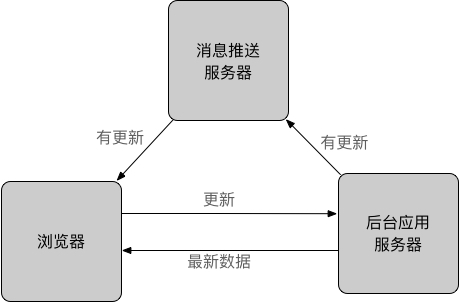
****网络优先和本地优先这两种策略是比较常见且容易实现的方案，而第三种“混合/ 智能”的策略是较难实现的。

图9离线数据更新示意图

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9所示。

## 3.4应用测试

由于每个浏览器内核的代码实现不一致，不仅渲染引擎对网页编写语法的解释不同，而且JS引擎对脚本语句的解释也是不同的，所以每个浏览器对Service Workers属性的支持程度也是不一样的。本节主要是对离线阅读应用在浏览器中进行兼容性测试。在五大浏览器平台上测试，浏览器列表如下表所示。主要是采取黑盒测试的方法，站在用户的角度，根据功能实际的操作流程，测试每个功能及功能按键。总结并分析测试结果，描述离线阅读应用的显示及功能是否正确。

**表4浏览器列表**

|  |  |  |
| --- | --- | --- |
| 名称 | 内核 | 特征 |
| IE | Trident | 只能应用于Windows平台，且是不开源 |
| FireFox | Geckos | 开源，以C++编写的网页排版引擎 |
| Chrome | Webkit | 开源，主要用于Mac OS系统，源码结构清晰、渲染速度极快 |
| Safari | Webkit | 开源，主要用于Mac OS系统，源码结构清晰、渲染速度极快 |
| Opera | Presto | 速度最快的引擎 |

该测试选择了当前比较流行的五大浏览器，分别在Windows、Mac OS操作系统上对离线数据的访问、提交和同步三大功能进行测试。测试结果如下表5所示。测试结果表明IE和Safari浏览器目前不支持，该离线阅读应用。

**表5浏览器兼容性测试结果**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 浏览器（版本） | 操作系统 | 离线数据访问 | 离线数据提交 | 离线数据同步 |
| Internet Explorer8 | WIN | × | × | × |
| FireFox | WIN | √ | √ | √ |
| Chrome  60.0.3112.113 | WIN | √ | √ | √ |
| Chrome | MAC | √ | √ | √ |
| Safari | MAC | × | × | × |
| Opera | WIN | √ | √ | √ |

## 3.5本章小结

本节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。

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。

第四章 PWP参考实现设计

本章主要对携式网络出版物PWP进行概述，通过对读者、出版商和作者、Web出版的打包，三个方面需求进行详细的分析。参考PWP搭建出整体的框架设计。

携式网络出版物PWP（Portable Web Publications），是W3C委员会为开放式的网络上定义一套数字出版的标准。W3C委员会为数字出版标准工作提出了便携式网络出版物技术挑战。其研究的主要目的是提出怎样让电子书阅读在浏览器中可以方便，便捷的实现。

2017年2月，W3C在与IDPF合并时正式设立了W3C数字出版计划（Publishing@W3C）。新成立的出版商务组（Publishing Business Group）作为出版业探讨对未来基于Web的数字出版主要需求的论坛，已经正式开始运行。新设立的 EPUB3 社区组（EPUB 3 Community Group）也开始正式运行，在W3C的框架下扩展 EPUB 规范。W3C的数字出版计划，使数字出版快速成为 W3C的一个新的行业领域。

2017年6月19日，W3C宣布设立了新的出版工作组（Publishing Working Group）。出版工作组的任务是：让所有的出版物成为Web上的第一类实体（first class entities）。工作组将为开放Web平台（Open Web Platform）提供必要的技术，确保传统出版与Web在无障碍访问、可用性、可移植性（便携性）、发行、归档、离线访问及可靠的交叉引用等方面的无缝融合。这是一个令人兴奋和雄心勃勃的目标，大量的W3C会员支持我们创建这个工作组，这也表明 W3C与IDPF的合并是具有战略意义的关键一步。作为一个标准化组织，W3C的工作成果是 Web标准，其最终目的是实现可互操作性（Interoperability）。

## 4.1技术挑战

目前，数字出版处于转折点。基于网络技术的EPUB被广泛应用于贸易书籍，并在教育界受到重视。但是创建EPUB文件不是那么容易的，需要专业的平台支持，然而一般的作者或者出版商是没有这样的条件的。并且渲染EPUB需要超出Web浏览器提供的大量额外的技术层，从而导致“epub阅读系统”的功能和质量差别很大。EPUB格式的数据文档已经满足不了开放式的Web环境。减少出版行业与网络之间的差距，将大大降低出版的复杂性，为作者提供方便的操作。

### 4.1.1网络出版物

网络出版物需要满足哪些条件才能达到便携式网络出版物的要求呢？实际上就是该出版物在网络上可以方便的传播，读者可以更方便地阅读。下面从如下几个方面进行分析。

1. 在线和离线

网络出版物的内容可以存储在本地存储或者缓存中。浏览器呈现引擎不用关心网络的状态，通过可编程的网络代理实现资源从本地存取，也可以从网络上更新。

1. 寻址和识别

URL可以唯一的标识一个网络或者资源。如果应用于网络出版物中，一个出版物应该有一个唯一的地址。例如，图书馆的图书标识，一个唯一的资源定位器，可以方便用户查找。所以，一个网络出版物也应该有一个或者几个标识符和一个定位器。

1. 文件清单

网络上电子出版物，其中包括HTML，CSS，IMG文件等一些修改性文件，也包括出版物的目录，文章等电子出版物详细内容的数据文件。还有其相关的作者，出版商等附加的数据。怎样把这些文件整合起来在网络中传输。

1. 元数据的发现

电子出版物的价格、作者、标题等。语法要支持Web搜索引擎。

1. 网络出版物的API

保证网络出版物格式，在其它应用程序中也可以使用。网络出版物是个集合文件，对于该集合作为一个整体来操作。如提供对网络出版的大纲或内容生成列表。

1. 样式、布局和分页

虽然在开放的网络平台上，并且CSS特性已有很大的提高，但仍不能够满足出版商期望的令人满意的印刷效果和布局数字出版解决方案。如渲染、数学和化学工式，必须由开放式的网络平台上提供技术来满足特别的格式要求。对于分页，在CSS或者DOM中的原生支持，可能需要网络出版物本地分页模式来定。

1. 显示控制和个性化

例如动态的改变字体的大小和背景颜色，还没有一个明确的基本框架来实现。个性化的显示控制。

以上是W3C数字出版物提出的便携式网络出版物的工作草案的基本内容。[24]其从宏观的方面进行了网络出版物的需求分析。下面从具体的需求用例对网络出版物进行分析。网络出版物用户群体包括“读者”、“出版商和作者”两大类。分别如下表4、5所示。

**表 6 读者的需求**

|  |  |  |  |
| --- | --- | --- | --- |
| PWP Usecase and Reqs: | Use cases | 支持/不支持 | 实现程度 |
| [Req. 1](https://www.w3.org/TR/2017/NOTE-pwp-ucr-20170502/#r_owp): Web出版物应该充分利用开方式网络平台OWP所提供功能 | 用HTML和CSS来表达内容 | V | 目前阅读支持 |
| 支持交互式显示，如互动考试 | V | 支持表单数据交互，如问卷调查或心理测试 |
| 无障碍、annotation的可移植（便携）、支持复杂的媒体格式（视频、flash动画、annimation） | V | 支持动画，多媒体的图书 |
| Req2: 有声读物，图表和混合媒体 | 图书内容中支持音视频的嵌入 | V | 支持背景音乐加入，开放制作页面加入音频 |
| Req 3: 包含数据 | 支持 CSV 格式 | X | 支持数据图表显示 |
| Req 4: 离线状态下可用 | 网络信号不好，离线时不用进行特定操作也可以阅读 | V | 基于Service Worker实现 |
| Req 5: 单一单位 | 搜索一个术语，不管书的内部结构是什么样的 | V | 实现以作者、内容搜索 |
| 设置书的字体、背景色等 | V | 字体大小、行间距等 |
| 页计数器，章节编号，脚注，尾注 | V |  |
| 辅助技术如屏幕阅读器或语音听写控制 | X |  |
| Req 6: 独特识别 | URL链接到出版物 | V | 支持在浏览器中阅读 |
| Req 7: 分页 | 阅读进展显示 | V | 纯文本的阅读有必要 |
| 翻页方便 | V |  |
| 翻页效果 | V | 翻页动画 |
| 页码固定 | V |  |
| Req 8: 个性化 | 对内容进行个性化设置 | V |  |
| 调整背景色 | V |  |
| 计算机代码显示 | V |  |
| 不显示无用的音频和视频内容 | X |  |

**表 7 出版商和作者的需求**

|  |  |  |  |
| --- | --- | --- | --- |
| PWP Usecase and Reqs: | Use cases | 支持/不支持 | 实现程度 |
| Req 1: 默认阅读顺序 | 按顺序排列每一章 | V |  |
| 能指出章节是按字母排序的 | X |  |
| Req 2: 随机访问内容 | 摘要 | V |  |
| 个性化定制，不同的读者，显示的内容不一样 | X | 用户选择阅读模式。  待定，有足够的时间就实现。 |
| 查出所有的图像 | X |  |
| Req 3: 组成资源信息 | 离线、低存储情况下，正常阅读 | V | 设置缓存数量 |
| 字体支持，数学公式 | X |  |
| 不重要的数据、部分内容可以不用离线缓存 | X |  |
| 提示数据更新 | X | 需要提示的可以设置提醒。 |
| Req 4: 访问控制和写保护 | 论文，相关资料控制访问。 | X |  |
| Req 5: 元数据和资源 | Web出版物中的一些附加信息 | V | 作者、摘要 |

### 4.1.2网络出版物的打包

1.出版物的格式

把Web出版物打包成一个文件，可以方便的进行传输、存储、共享或归档。目前存在几种打包格式，使用ZIP和XML清单一起使用。例如EPUB的OCF,办公文档的ODF或者OOXML。但是在文档发布平台或Web环境中，这些打包格式还没有得到普遍认可和支持。打包不与Web出版物的体系结构紧密耦合，那样打包格式就不能用于不同的环境下面。

2.安全模型

网络的安全模型，主要基于同源策略和“网站”的概念，不适用于便携式文件，因为“源”的概念基于HTTP属性，文档从其在线状态转换到便携状态时，该属性无效/不存在。（打包）Web出版物必须包含一个安全模型，该模型定义在线和便携状态的规则，并与Web包装相一致。

**表 8 打包的WEB出版物**

|  |  |  |  |
| --- | --- | --- | --- |
| PWP Usecase and Reqs: | Use cases | 支持/不支持 | 实现程度 |
| Req 1: 打包的Web出版物的分发不应影响其迭代 | 可以通过邮件分发、通过SD卡和蓝牙传输 | V | 支持URL访问，打包成epub |
| Req 2: 分布和迭代 | 可以跨终端显示出版物 | V |  |
| 个人添加的标记互不影响 | V |  |
| 可以查看出版物的阅读量数据 | X |  |
| Req 3: 分配过程 | 打包出版物并分发 | V |  |
| Req 4: 在组件中定位元数据 | 描述性元数据文件 | V |  |
| Req 5: 新建或删除的组件 | 发现组件删除或新增 | X |  |
| Req 6: 重映射链接 | 重新映射连接 | X |  |
| Req 7: 诚信认证 | 出版商证明出版物在交付期间没有被篡改 | X |  |
| Req 8: 出版物源的真实性 | 出版商提供可用于检查出版物的来源及其真实性的信息 | X | 文章里加入来源信息 |
| Req 9: 不断升级 | 提供用于升级特定出版物的信任的方法 | X |  |

## 4.2 PWP框架总体设计

从上述的技术挑战需求中，可以分析出，实现PWP的实际意义在于，在网络中形成数字阅读的统一标准。读者、作者和出版商对网络出版物的要求。网络出版物的打包应该满足哪些条件。

PWP架构设计图如下图10所示。

## 4.3 本章小结

图10 PWP框架设计图

本章节主要对携式网络出版物PWP进行概述，并对现有的技术挑战进行了介绍。从网络出版物的内容格式到它的打包形式，都进行了详细的分析。对每一个可能存在的用例需求进行了描述。最后参考PWP，完成实现了PWP框架的总体设计。

第五章 书刊发布阅读系统实现

传统出版物的发布要经历以下流程，作者提供内容，编辑设计排版，印刷成册，销售发行。其中涉及到的人员，包括作者、编辑、印刷工人，销售人员。这个过程不仅浪费人力资源，也浪费物质资源。过程繁琐，时间周期长。

## 5.1 需求分析

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大，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]。

同时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，如阅读文章时做问卷题，写书评等。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；但另一方面，该交互在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，用必须下载安装相关的应用APP，且该APP支持离线交互功能；另一种离线体验就是，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。第一种方式，让用户安装APP应用，增加用户的学习成本，占用手机空间。第二种方式，无法实现用户交互体验，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，以及浏览器产业的支持。而且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。[12]

随着Service Worker的到来，新型的离线阅读使其在Web上实现原生应用成为了可能。它给本地阅读带来简便、快速、offline的新体验，Service Worker可以将文章缓存到本地，可以离线阅读，方便操作管理。

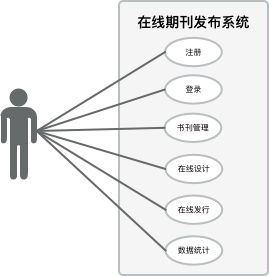
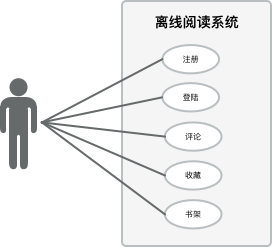
该系统的设计即要符合传统电子书刊出版的流程，也要兼顾数据阅读的特性，充分发挥互联网的优势。后台管理在线书刊发布系统用例图如下图12左图所示。

图 11 系统用例图

文章在发布系统中编辑发布后，读者就可以在离线阅读系统中进行阅读。离线阅读系统用例图如10右图所示。

## 5.2 概要设计

本节在需求分析的基础上对系统进行概要设计，首先对系统进行总体架构设计，再对系统进行功能模块的划分，最后根据功能模块设计数据库表。

### 5.2.1 总体架构设计

图 12 系统整体架构图

互联网的发展，网络阅读成为优于传统纸张阅读的新型阅读方式。图书保存于网络，一点即可阅读。用户可利用书刊发布阅读系统，制作图书，并保存在本系统数据库中。发布后，读者打开客户端即可进行阅读。书刊发布阅读系统充分利用互联网开发技术，以实现用户阅读手机端舒适的阅读体验；用户内容制作PC端方便高效的发布体验。系统涉及到两类用户，一个是内容的使用者（读者），另一个是内容的制作者（作者）。作者制作内容，提供数据源，供读者阅读。系统的整体架构图，如下图11所示。

系统采用B/S(浏览器/服务器)，架构设计。读者不用安装客户端，可以直接在手机浏览器中进行阅读，不占用读者手机端空间。编辑人员可以随时随地，在电脑浏览器中打开网页直接进行内容的编辑制作，不管在何地，都能保证内容的同步。B/S架构，系统升级和维护成本低，开发人员只需更新服务器端代码，就可以升级系统。

服务器端采用基于node.js平台的Express开发框架，并实现MVC模式的代码架构设计。模块化的设计，降低了代码的耦合性，提高了代码日复用率，保证了产品高质量开发。

图 13 系统分层架构图

以层次化的结构设计系统，能使项目代码结构简结清晰，分层架构图，如下图所示。用户接入层包括移动端阅读用户和PC端编辑发布用户。视图层展示包括阅读页面、书刊列表、书刊管理、在线设计、在线发行。请求控制层用统一的访问控制系统，通过router对每个请求进行控制，根据相关的业务逻辑，取相关数据返回到前端视图层。后台服务上的数据源，包括图片服务器、应用服务器、关系数据库和Redis服务。

系统架构的技术方案如下：

1、技术语言

在Node.js运行环境下，使用Javascript编写，，依赖于Chrome V8引擎进行代码解释。其特点是事件驱动，非阻塞I/O，轻量、可伸缩，更适于实时数据交互应用。

2、数据库

数据库采用postgreSQL。PostgreSQL 的特性覆盖了 SQL-2/SQL-92 和 SQL-3/SQL-99，是目前世界上支持最丰富的数据类型的数据库。其支持事务、子查询、多版本并行控制、数据完整性检查等特性的数据库管理系统。

3、数据缓存

缓存采用Redis，其读写性能优异，支持数据持久化，支持主从复制，可以进行读写分离。数据结构丰富。

4、模板引擎

模板引擎采用Express，默认支持的ejs，与系统框架整合度更高，集成更好。

5、系统框架

使用流行框架Express，特点是高性能，Express仅在web应用相关的nodejs模块上进行了适度的封装和扩展，较大程度避免了过度封装导致的性能损耗，而且扩展性强，基于中间件的开发模式，使得express应用的扩展、模块拆分非常简单，既灵活，扩展性又强。

6、前端框架

Vue 只关注视图层， 采用自底向上增量开发的设计。Vue 的核心库只关注视图层，它不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与单文件组件和 Vue 生态系统支持的库结合使用时，Vue 也完全能够为复杂的单页应用程序提供驱动。

### 5.2.2 功能模块划分

书刊发布阅读系统的功能模块图如图13所示。系统分两大类功能模块。一部分是书刊发布系统，其中包括，书刊管理、在线设计、在线发行、注册、登录功能模块。另一部分是书刊阅读系统，其中包括，书刊阅读、书刊评论、书刊搜索、注册和登录功能模块。注册和登录功能在两个子系统中为公用功能模块，后台代码逻辑公用一套****实现。

图 14 系统功能模块图

### 5.2.3 数据库设计

数据库中数据表关系图如下图14所示。从图中可以看出文章表是最为重要的数据库表。书的相关信息存在book表中，一本book可能有若干个期刊，期刊信息存放在issue表中，期刊的栏目信息存放在article表中，期刊的文章信息存放在page表中。client表中存放客户信息，一个客户可能有多个用户，用户信息存于reviewer表中。template\_type存放设计样式类型。template\_pub表放可用样式。

因离线应用需要一个强大并且可靠的数据库，因此本系统数据库选择的是以单一指责标准化原则设计的。而且为了满足SOLID原则，要运用一些常见的OOP原则（如单一责任原则、接口隔离、依赖反转）去改进数据模型和数据库代码。并涉及到了对象关系特性（如表继承），只有类似PostgreSQL的数据库才有这些特性。因此数据库选择使用PostgreSQL。

SOLID原则即为一个对象应该只包含单一的职责，并且该职责被完整地封装在一个类中。（Every object should have a single responsibility, and that responsibility should be entirely encapsulated by the class.），即又定义有且仅有一个原因使类变更。这样的优点是降低类的复杂性，类的职责清晰明确。比如数据职责和行为职责清晰明确。提高类的可读性和维护性，并且使变更引起的风险减低，变更是必不可少的，如果接口的单一职责做得好，一个接口修改只对相应的类有影响，对其他接口无影响，这对系统的扩展性、维护性都有非常大的帮助。

Postgresql体系结构和进程视图，如图15所示。

**图 15 数据表关系图**

## 5.3 详细设计与实现

在线书刊发布系统包括用户注册、登录、书刊管理、在线设计、在线发行、数据统计，帮助论坛这七大功能模块。

离线阅读系统包括个人用户注册、登录、评论、收藏、书架功能模块。主要是用Service Worker实现离线功能。

分书刊发布系统和离线阅读系统的详细设计与实现。

### 5.3.1 书刊发布系统

1 注册登录模块

编辑人员，需要登录书刊发布系统，用可以进行书刊的制作发布操作了。这个模块可以非常简单，填下用户名密码就可以了，但是如今账户信息的重要性越来越大，安全性更是一个重要的因素，这其中隐藏着很多逻辑陷阱，所以在登陆注册系统的设计中也是一个便捷性与安全性的取舍平衡。

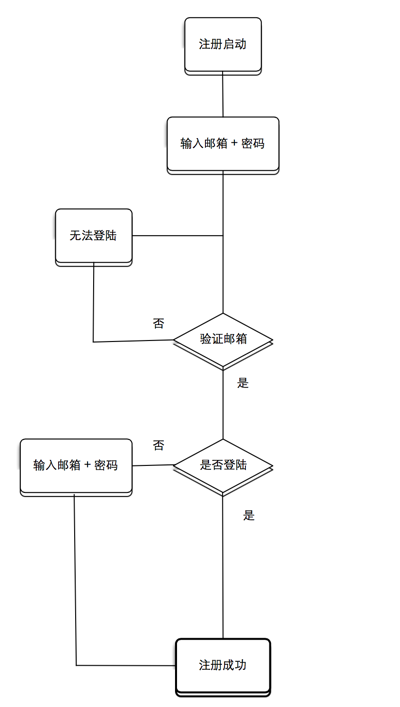
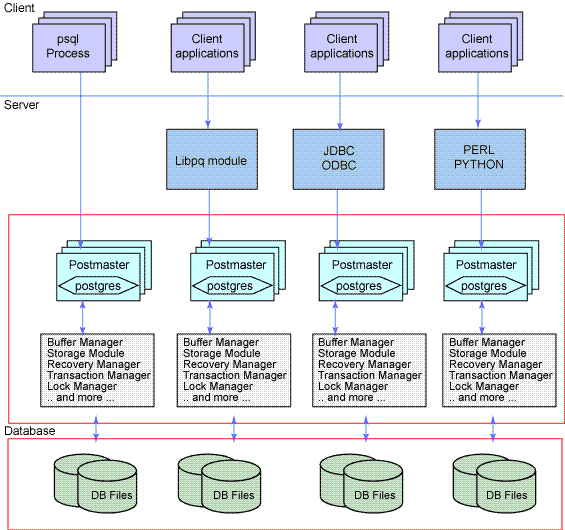
在该系统中自建用户系统,用户需要用邮箱登录注册。注册时，系统发送验证码到邮箱，验证成功后，登录系统。

图 16 Postgresql体系结构和进程视图

2 书刊管理模块

书刊管理模块的供编辑人员对书刊进行新增、修改和删除操作。同时也可以设置书刊的相关信息，如书刊的作者，是否使用栏目等信息。使用栏目的书刊，可以对栏目进行管理。

2 在线设计模块

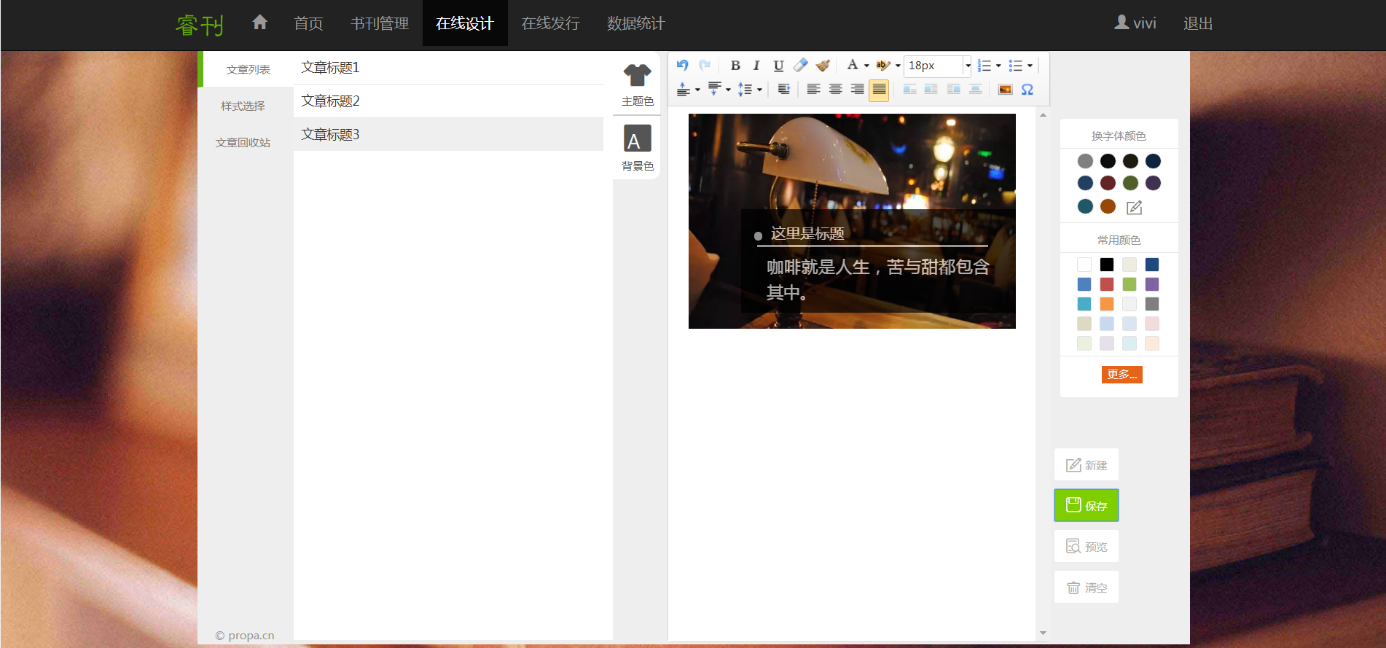
该模块是本系统中核心模块之一。本模块的主要功能是对文章样式进行编辑设计。Ueditor提供文章的常见编辑功能，同时系统中提供各种样式模板，如标题、正文、图文，页眉和分隔线的常用样式，对文章样式的编辑都是一键操作，可以修改文章字体色、背景色，也可修改文章主题色。编辑人员不用复杂的编辑制作工具，仅用该模块就可以编辑制作出精美的，有设计感的文章排版。实现效果图如下图16。

图 17 在线设计实现效果图

3 在线发行模块

在线发行模块，完成书刊的编辑发行。在线设计完成的文章，在该模块的文章列表中显示出来，用户选中一本书刊，将文章拖入到发布列表中。点击立即发行，完成书刊的发行操作。发行完成后，读者可以在网络上通过链接直接打开该书刊阅读，也可以在书城列表中搜索到该书刊。同时用户也可以取消发行书刊。

### 5.3.2 离现阅读系统

1 阅读模块

阅读模块，是该系统中核心的模块之一。用户不需要登录，在手机浏览器中打开一本书的链接，翻页阅读，同时也可以把该书加入到书架。同时支持用户离线阅读，用户在有网的时候，打开阅读，系统自动缓存文章列表。在网络信息差和没网的时候，同样可以阅读。也可下载出EPUB格式的电子书，用其它软件或者设备阅读。

2 书刊列表

书刊列表，分书架和发现两类书刊列表。书架中放的是用户自己收藏的书刊列表。发现中，用户可搜索书刊，发现自己感兴趣的书刊。

## 5.4 系统测试

便携式网络出版物PWP（Portable Web Publications），是W3C委员会为开放式的网络上定义一套数字出版的标准。W3C委员会为数字出版标准工作提出了便携式网络出版物技术挑战。其研究的主要目的是提出怎样让电子书阅读在浏览器中可以方便，便捷的实现。本节对便携式网络出版物的需求进行分析，并以此为参考实现，并测试。

### 5.4.1 功能测试

功能测试是对产品的各功能进行验证，根据功能测试用例，逐项测试，检查产品是否达到用户要求的功能。本模块中重点介绍离线阅读模块的测试用例，和测试结论。测试环境，软件Chrome 。

**表 9 书刊阅读部分测试用例**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **编号** | **测试目的** | **前序条件** | **输入操作** | **期望结果** | **测试结果** |
| it-001 | 页面展示 | 用户通过浏览器进入首页 | 输入网址 | 进入页面，书架中显示收藏的图书。当没有收藏图书时，显示发现页面 | 通过 |
| it-002 | 页面切换 | 进入首页 | 点击底部书架\发现 | 点击书架，显示自己收藏的书刊表表，点击发现显示所有书刊列表 | 通过 |
| it-003 | 阅读页面展示 | 进入首页 | 点击书刊列表中的书 | 打开对应的书详细页面，可翻页进行阅读 | 通过 |
| it-004 | 搜索 | 进入首页-发现 | 输入关键字 | 根据输入关键字查询出相关书刊列表 | 通过 |
| it-004 | 目录显示 | 阅读书刊页面 | 单击屏幕 | 屏幕下方显示导航条，点击列表图标，显示当前书的目录列表，点击目录跳转到对应的文章页 | 通过 |
| it-005 | 收藏 | 阅读书刊页面 | 单击屏幕 | 屏幕下方显示导航条，点击收藏图标，收藏成功，书架中显示该书刊 | 通过 |
| it-006 | 下载 | 阅读书刊页面 | 单击屏幕 | 屏幕下方显示导航条，点击下载图标，下载书刊为EPUB格式到本地 | 通过 |
| it-007 | 设置阅读样式 | 阅读书刊页面 | 双击屏幕 | 屏幕下方显示样式设置导航条，可设置背景色，字体大小，行间距。双击屏幕，关闭导航条。 | 通过 |
| it-008 | 离线阅读 | 阅读书刊页面 | 第二次打开页面 | 关闭网络连接，数据返回正常 | 通过 |

测试结果通过。

### 5.4.2 兼容性测试

### 5.4.3 性能测试

## 5.4 本章小结

书刊发布阅读系统包括内容制作平台和内容阅读平台。用户可以在平台上完成一体化操作，不用关注电子刊物的格式，也不用担心其在网络上怎么传播，只用关注内容和简单的排版制作。编辑完成后，系统自动生成可阅读的链接地址。用户打开地址就可以进行阅读。

本章详细描述了系统的设计与实现。并完成相关功能测试。

第六章 结论与展望

## 6.1 论文总结

基于Service Worker技术，设计与实现了离线阅读系统。并与书刊发布系统进行整合，实现书刊发布阅读系统。参考PWP框架需求，进行功能测试。本文的主要工作如下：

1. 介绍选题的背景与意义，分析了国内外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。

## 6.2 展望

本文中虽然实现了基于Service Worker离线阅读系统，但是该系统还存在着如下一些需要进一步改进的方面：

1．基于Service Worker离线阅读系统依赖于浏览器软件，一小部分不支持Service Worker新特性的浏览器，也不支持离线功能。

2．对离线数据的同步，运用的是拉取实现，不是真正意义上推送实现。如果有需要可以单独搭建推送平台，实现推送服务。

3．W3C委员会提出的携式网络出版物，文档内容只是草案，随时可能会变动，需求用例有些也不明确。有待进一步讨论。

参考文献

1. 马军. 电子书是否可以取代传统阅读[J]. 文化纵横谈，群文天地，2013年第3期下.
2. 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).
3. 姚柏年. 数字出版商业模式研究[D]. 华东师范大学， 2012.
4. 邱珊. 使用HTML5WebWorker提高Web的应用性能研究 [J]. 软件导刊，2013(12).
5. 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).
6. 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).
7. 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)
8. 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)
9. 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).
10. 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).
11. 章斓. 基于HTML5的Web App的开发与探索[J]. 长沙大学学报，2015(05).
12. 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)
13. 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).
14. 邱珊. 使用 HTML5 WebWorkers提高 Web 的应用性能研究[J]．软件导刊.2013(12).
15. (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
16. 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.
17. 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.
18. MarcosCaceres，KennethRohdeChristiansen，MounirLamouri，AnssiKostiainen.Web App Manifest[EB/OL]，https://w3c.github.io/manifest/，2013-12-17/2016-12-12
19. 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
20. 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
21. 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.
22. 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)
23. 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
24. 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
25. 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)
26. Ivan Herman， W3C.Portable Web Publications: Technology Challenges [EB/OL]. https://www.w3.org/2016/Talks/W3CTrack-IH/， 2016-04-13
27. Matt Gaunt. Service Workers: an Introduction[EB/OL]. https://developers.google.com/web/fundamentals/getting-started/primers/service-workers， 2016-12-21
28. Sights.TheHTML5test[EB/OL].http://html5test.com/results/desktop.html， 2013-3-20
29. WhatWG. HTML5 living standard[EB/OL]. http://www.whatwg.org/specs/web-apps/current-work/multipage/，2016.12
30. W3schools．HTML5 教程[EB/OL].<http://www.w3school.com.cn/html5/index.asp，2016.12>
31. 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。本论文从选题、方案设计、到最终论文的完成，胡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。胡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。在这里我向胡教授表示深深的谢意和良好的祝愿。

其次感谢班主任老师和工程硕士的所有同学们，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。

最后衷心感谢所有关心、爱护、帮助过我的亲人和朋友们。