电子科技大学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

综合课程设计论文

论文题目:WEb新闻发布系统

专　业:计算机科学与技术

学　号:2016060107017

作者姓名:张国伟

指导教师:赵勇

摘要

这是一个最好的时代，也是最坏的时代，我们亲身经历着激动人心的变革，也往往会陷入选择的迷茫。随着浏览器版本的革新与硬件性能的提升，Web 前端开发进入了高歌猛进，日新月异的时代，无数的前端开发框架、技术体系争妍斗艳，让开发者们陷入困惑，乃至于无所适从。特别是随着现代 Web 前端框架(Angular 、 React、Vue.js)的出现，JavaScript 、 CSS、HTML 等语言特性的提升，工程化、跨平台、大前端等理论概念的提出，Web 前端开发的技术栈、社区也是不断丰富完善。本文主要是关于`React`, `express` , `mongodb`的使用,探究与思考.

关键词 : `React` `node.js` `mongodb` 组件化

第一章 绪论

1.1 研究工作的背景与意义

在Web开发中，我们总需要将变化的数据实时反应到UI上，这时就需要对DOM进行操作。而复杂或频繁的DOM操作通常是性能瓶颈产生的原因（如何进行高性能的复杂DOM操作通常是衡量一个前端开发人员技能的重要指标）。因为如此`React`横空出世.

伴随着前端的飞速进步,全栈,大前端等概念逐渐成为主流,而 `node.js`的突然出现大大给了前端开发者一个了解后端的机会.

Node.js是一个可以快速构建网络服务及应用的平台。该平台的构建是基于Chrome'sJavaScriptruntime，也就是说，实际上它是对GoogleV8引擎(应用于GoogleChrome浏览器)进行了封装。Nodejs不是一个js应用而是一个js运行平台。其是由C++编写而成。但是Nodejs是一个后端的运行环境。因此你可以编写系统级或者服务器端的js让Nodejs帮你执行。node.js可以用js来写,降低了前端开发者学习后端的成本. [1]

近年来,NoSQL的概念也蓬勃发展.大大小小的Web站点在追求高性能高可靠性方面，不由自主都选择了NoSQL技术作为优先考虑的方面。mongodb正是其中的代表.

这些新技术,新事物的出现,一方面印证了前端方兴未艾,另一方面这些新事物的出现也正大力推进着前端的进步与发展.在这样的背景下,学习这些新事物的重要性不言而喻.本论文便是从头开始实现一个基于`React`, `express` , `mongodb`的WEB新闻发布系统.借此学习了解这些新事物.

1.2 前端的历史与现状

任何一个编程生态都会经历三个阶段，首先是原始时期，由于需要在语言与基础的 API 上进行扩充，这个阶段会催生大量的辅助工具。第二个阶段，随着做的东西的复杂化，需要更多的组织，会引入大量的设计模式啊，架构模式的概念，这个阶段会催生大量的框架。第三个阶段，随着需求的进一步复杂与团队的扩充，就进入了工程化的阶段，各类分层 MVC，MVP ， MVVM 之类，可视化开发，自动化测试，团队协同系统；这个阶段会出现大量的小

而美的库。我们可以简要地将 JavaScript 出现以来至今的发展历史划分为以下时代:

Era/ 时代	Timeline/ 时间线	Problems/ 问题	Innovations/ 创新	Dominant Browsers/ 主流浏览器
蛮荒时代	大概 1996 – 2004	基础 DOM 操作，用户交互	JavaScript 本身，XHR 与 AJAX	Netscape Navigator, Microsoft Internet Explorer
jQuery 时代	大概 2004 – 2010	增长的网页复杂度，大量的浏览器兼容需求	健壮的 DOM 操作，早期的 SPA 单页应用	Microsoft IE，Mozilla Firefox
SPA 时代	大概 2010- 2014	DHTML 过载，大规模数据操作，性能	MVC 框架，双向数据流，DOM 自动化	Google Chrome，Microsoft IE，Mozilla Firefox，Apple Safari
现代	大概 2014- 现在	性能，复杂应用的状态管理，可用性	Virtual DOM，单向数据流，类型系统，测试	Google Chrome，Apple Safari

Web 前端开发可以追溯于 1991 年蒂姆 · 伯纳斯 - 李公开提及 HTML 描述，而后 1999 年 W3C 发布 HTML4 标准，这个阶段主要是 B/S 架构，没有所谓的前端开发概念，网页只不过是后端工程师的顺手之作，服务端渲染是主要的数据传递方式。接下来的几年间随着互联网的发展与 REST 等架构标准的提出，前后端分离与富客户端的概念日渐为人认同，我们需要在语言与基础的 API 上进行扩充，这个阶段出现了以 jQuery 为代表的一系列前端辅助工具。2009 年以来，智能手机开发普及，移动端大浪潮势不可挡，SPA 单页应用的设计理念也大行其道，相关联的前端模块化、组件化、响应式开发、混合式开发等等技术需求甚为迫切。这个阶段催生了 Angular 1、Ionic 等一系列优秀的框架以及 AMD、CMD 、 UMD 与 RequireJS、SeaJS 等模块标准与加载工具，前端工程师也成为了专门的开发领域，拥有独立于后端的技术体系与架构模式。

而近两年间随着 Web 应用复杂度的提升、团队人员的扩充、用户对于页面交互友好与性能优化的需求，我们需要更加优秀灵活的开发框架来协助我们更好的完成前端开发。这个阶段涌现出了很多关注点相对集中、设计理念更为优秀的框架，譬如 React、Vue.js 、 Angular 2 等组件框架允许我们以声明式编程来替代以 DOM 操作为核心的命令式编程，加快了组件的开发速度，并且增强了组件的可复用性与可组合性。而遵循函数式编程的 Redux 与借鉴了响应式编程理念的 MobX 都是非常不错的状态管理辅助框架，辅助开发者将业务逻辑与视图渲染剥离，更为合理地划分项目结构，更好地贯彻单一职责原则与提升代码的可维护性。在项目构建工具上，以 Grunt、Gulp 为代表的任务运行管理与以 Webpack、Rollup 、 JSPM 为代表的项目打包工具各领风骚，帮助开发者更好的搭建前端构建流程，自动化地进行预处理、异步加载、Polyfill 、压缩等操作。

JavaScript 虽匆匆而生却自由生长，历经，在 ES6 之后；在 TypeScript、Flow 等静态类型语言的。最负盛名的当是 Chrome 内置的 V8 引擎，也是 Node.js 的基石. [2]

1.3 本论文的主要探究

本论文主要探究`React` 和 `node.js` `mongodb` 配合使用的实践,由于时间较短,并没有花更多时间去写css样式,专注于组件化的`React` ,后端使用了 `node.js` 运用最广的 `express` 框架 ,并使用`mongoose` 来连接`node.js`和`mongodb`.

1.4 本论文的结构安排

本论文章节安排如下:

第一章 综合论述本文的的内容,方法和组织

第二章 论述改项目中用到的基础知识

第三章 结合项目的代码论述项目如何实现

第四章 分析项目结果及展望后续工作

第二章 课程设计实现基础

2.1 node.js

Node.js起源于2009年3月.最初Ryan Dahl是为了构建一个高性能的文本服务器,而node.js是他在寻找一种更高提升web服务器性能发现的，事件驱动、非阻塞I/O这种方式能更好的提高性能.随着Nodejs的不断发展，Nodejs渐渐演变成一种构建网络应用的基础框架，并发展为一个不共享任何资源的单线程、单进程系统，但包含了很适合网络的库，这样Nodejs就为构建大型分布式应用提供了基础设施.他们的目标都是为了构建快速、可伸缩的网络应用平台。它自身非常简单、采用通信协议来组织许多的Node，非常容易通过拓展来达成构建大型网络应用的目的.Node.js是一个可以快速构建网络服务及应用的平台。该平台的构建是基于Chrome'sJavaScriptruntime，也就是说，实际上它是对GoogleV8引擎(应用于GoogleChrome浏览器)进行了封装。Nodejs不是一个js应用而是一个js运行平台。其是由C++编写而成。但是Nodejs是一个后端的运行环境。因此你可以编写系统级或者服务器端的js让Nodejs帮你执行。

node.js具有一个非常良好的社区生态,大量的包被发布在npm上供开发者使用,这是node.js的优点,也是它的缺点,另外node.js具有以下优点

1. node.js采用事件驱动、异步编程，为网络服务而设计。
2. node.js采用非阻塞模式的IO处理，可以使node.js在相对低系统资源耗，拥有出色的负载能力，非常适合用作依赖其它IO资源的中间层服务。
3. node.js轻量高效，可以认为是数据密集型分布式部署环境下的实时应用系统的完美解决方案。

2.2 mongodb

MongoDB 是一个基于分布式文件存储的数据库。由 C++ 语言编写。旨在为 WEB 应用提供可扩展的高性能数据存储解决方案。MongoDB 是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。

近年来NoSQL之所以可以大热,是因为它具有已下优点:

1. 易扩展

   NoSQL数据库种类繁多，但是一个共同的特点都是去掉关系数据库的关系型特性。数据之间无关系，这样就非常容易扩展。也无形之间，在架构的层面上带来了可扩展的能力。

2. 大数据量,高性能

NoSQL数据库都具有非常高的读写性能，尤其在大数据量下，同样表现优秀。这得益于它的无关系性，数据库的结构简单。一般MySQL使用Query Cache，每次表的更新Cache就失效，是一种大粒度的Cache，在针对web2.0的交互频繁的应用，Cache性能不高。而NoSQL的Cache是记录级的，是一种细粒度的Cache，所以NoSQL在这个层面上来说就要性能高很多了。

3. 灵活的数据模型

   NoSQL无需事先为要存储的数据建立字段，随时可以存储自定义的数据格式。而在关系数据库里，增删字段是一件非常麻烦的事情。如果是非常大数据量的表，增加字段简直就是一个噩梦。这点在大数据量的web2.0时代尤其明显。[3]

2.3 React

React为了实现高性能的复杂DOM操作引入了虚拟DOM（Virtual DOM）的机制：在浏览器端用Javascript实现了一套DOM API。基于React进行开发时所有的DOM构造都是通过虚拟DOM进行，每当数据变化时，React都会重新构建整个DOM树，然后React将当前整个DOM树和上一次的DOM树进行对比，得到DOM结构的区别，然后仅仅将需要变化的部分进行实际的浏览器DOM更新。尽管每一次都需要构造完整的虚拟DOM树，但是因为虚拟DOM是内存数据，性能是极高的，而对实际DOM进行操作的仅仅是Diff部分，因而能达到提高性能的目的。

基于HTML的前端界面开发正变得越来越复杂，其本质问题基本都可以归结于如何将来自于服务器端或者用户输入的动态数据高效的反映到复杂的用户界面上。`React`框架正是完全面向此问题的一个解决方案，按官网描述，其出发点为：用于开发数据不断变化的大型应用程序（Building large applications with data that changes over time）。相比传统型的前端开发，React开辟了一个相当另类的途径，实现了前端界面的高效率高性能开发。

React具有以下特性

1. 声明式

   React 可以非常轻松地创建用户交互界面。为你应用的每一个状态设计简洁的视图，在数据改变时 React 也可以高效地更新渲染界面。以声明式编写UI，可以让你的代码更加可靠，且方便调试。

2. 组件化

   创建好拥有各自状态的组件，再由组件构成更加复杂的界面。无需再用模版代码，通过使用JavaScript编写的组件你可以更好地传递数据，将应用状态和DOM拆分开来。

3. 一次学习，随处编写

   无论你现在正在使用什么技术栈，你都可以随时引入 React 开发新特性。React 也可以用作开发原生应用的框架 React Native.

React是一个全新思路的前端UI框架，它完全接管了UI开发中最为复杂的局部更新部分，擅长在在复杂场景下保证高性能；同时，它引入了基于组件的开发思想，从另一个角度来重新审视UI的构成。通过这种方法，不仅能够提高开发效率，而且可以让代码更容易理解，维护和测试。Facebook以这样一种方式将沉淀多年的前端开发经验和技术的积累完全开源出来，值得所有前端开发者去借鉴和学习。并且React在发布五年的时间里就获得了极大的关注，Github上拥有将近10万的Star，相信其对前端开发的方向，甚至Web Component的标准，都将产生一定的影响。

第三章 课程设计具体代码实现

3.1 mongoose结构

```
const user=new Schema({
  // 用户名
  user:{
    type: String,
    require: true
  },
  // 密码
  pwd:{
    type: String,
    require: true
  },
  // 用户注册时间
  time:{
    type: String,
    require: true
  },
  // 用户头像
  avatar:{
    type: String
  },
  // 用户简介
  desc:{
    type: String
  },
  // 用户权限,用来控制是否有发帖权限
  permission:{
    type: Boolean,
    require: true,
    default: false
  }
});
const news=new Schema({
  // 标题
  tittle:{
    type: String,
    require: true
  },
  // 新闻主体内容
  content:{
    type: Object,
    require: true
  },
  // 新闻发布时间
  time:{
    type: String,
    require: true
  },
  // 新闻的主图
  img:{
    type: String,
    require: true
  },
  // 新闻的类型
```

```
  type:{
    type: String,
    require: true
  },
  // 新闻发表者用户名
  user:{
    type: String,
    require: true
  },

  // 评论
  comments:{
    comment:{
      type: String,
      require: true
    },
    // 评论发表时间
    time:{
      type: String,
      require: true
    },
    // 评论发表者用户名
    fromuser:{
      type: String,
      require: true
    },
    // 被回复者用户名
    touser:{
      type: String,
      require: true
    },
  }
})
```

3.2 后端express代码

```
//由于代码比较多,不全部贴出,只贴出具有代表性的部分
const fs = require('fs');
const express = require ('express');
// 引入 multer 库 ,用来保存本地图片到服务器
const multer  = require('multer');

// 最终新闻主图和头像会分别保存在 service/uploadimg 和service/uploadavatar 中
const uploadimg = multer({ dest: 'uploadimg/' })
const uploadavatar = multer({ dest: 'uploadavatar/' })

//引入 mongodb 数据模型
const {User,News} = require('./model.js');

//引入body解析器,由于后来的express删除了这个模块,所以需要手动引入
const bodyParser = require('body-parser');
```

```
//解析cookie
const cookieParase = require('cookie-parser');

const app = express();
// express限制文件大小50m
app.use(bodyParser.json({limit: "50mb"}));
//使用cookie解析的中间件
app.use(cookieParase());

//下面是主体逻辑,包含登录,注册,发表,删除,修改等,这里只列出注册账号部分和上传头像部分

//上传头像
app.post('/upavatar', uploadavatar.single('file'),function(req,res){
  //如果上传的头像格式是图片并且大小小于5m
  if(req.file.mimetype.slice(0,5)==="image"&&req.file.size<5000000){
    return res.json({
      code: 0,
      msg: "上传头像成功! ",
      avatarname:req.file.filename
    });
  }else {
    return res.json({
      code: 1,
      msg: "不支持该格式,或者图片大小大于5M,请重新上传! "
    });
  }

})

//注册帐号
app.post('/register',function(req,res){
  const userinfo=req.body;
  User.findOne({user:userinfo.user},function(err, doc){
    if(doc){
      return res.json({
        code: 1,
        msg: "这个用户名已经被注册了! "
      });
    }
    //如果该账号没有被注册,在数据库中创建这个账户
    User.create(userinfo, function (err, doc) {
      if (err) {
        return console.log('err',err);
      }
    });
    return res.json({
      code: 0,
      msg: "注册成功! "
    });
  })
})


//监听9003端口
app.listen(9003,function(){
  console.log("ok 9003");
})
```

3.3 前端react代码

```
//这部分代码虽然不尽相同,单整体套路相似,下面贴出发表文章页面的component代码
import React from 'react';
import './publish.css'
import axios from "axios";
import E from 'wangeditor'
class Publish extends React.Component{
  constructor(props){
    super(props)
    this.state={
      title:"",
      content:"",
      time:"",
      img:"news.png",
      type:"",
      user:""
    }
  }
  handleImgChange () {
    const registerInfo=document.querySelector(".registerInfo")
    const input=document.querySelector('#imgfile');
    const img=document.querySelector('.newsMainImg');
    const form=document.querySelector('.imgForm');
    const fd = new FormData(form);

    // 下面三行是主图预览,上传图片后在本地预览头像
    let reader = new FileReader();
    reader.onload = (function(aImg) { return function(e) { aImg.src =
e.target.result; }; })(img);
    reader.readAsDataURL(input.files[0]);

    //上传本地头像到服务器,并把服务器返回的图片文件名保存到state中
    axios.post("/upimg",fd).then(res => {
      if (res.status === 200 && res.data.code === 0) {
        this.setState({img:res.data.imgname})
        //显示是否上传头像成功
        registerInfo.style.display='block';
        registerInfo.innerHTML=res.data.msg;
      } else {
        registerInfo.style.display='block';
        registerInfo.innerHTML=res.data.msg;
      }
    });
  }
  componentDidMount(){

    // E 初始化编辑器
```

```
    const elem = this.refs.editorElem
    const editor = new E(elem)
    editor.customConfig.uploadImgShowBase64 = true   // 使用 base64 保存图片
    // 使用 onchange 函数监听内容的变化，并实时更新到 state 中
    editor.customConfig.onchange = html => {
      this.setState({
        content: html
      })
    }
    editor.create()
    // E
  }
  //把this.State中的新闻内容发送到后端,让后端保存到mongodb数据库中
  clickHandle() {
    const registerInfo=document.querySelector(".registerInfo")
    const time=new Date().toLocaleString();
    const obj = document.querySelector(".selectType"); //定位id
    const index = obj.selectedIndex; // 选中索引
    const type = obj.options[index].text; // 选中文本
    const user=document.cookie.slice(4);
    const {content,img}=this.state;
    const tittle=document.querySelector(".publishInputTitle").value
    console.log({time,content,user,type,img,tittle})
    axios.post("/publish", {time,content,user,type,img,tittle}).then(res => {
      if (res.status === 200 && res.data.code === 0) {
        //显示是否发表成功
        registerInfo.style.display='block';
        registerInfo.innerHTML=res.data.msg;
      } else {
        registerInfo.style.display='block';
        registerInfo.innerHTML=res.data.msg;
      }
    });
}
  render(){
    return(
      {/*主体内容*/}
      <div className="publishInput">
        <form className="imgForm" action="/uploadimg" enctype="multipart/form-
data">
          <input type="file" name="file" id="imgfile" onChange={()=>
{this.handleImgChange()}}/>
        </form>
        <img src="https://s1.ax1x.com/2018/05/26/ChiaHf.png"
className="newsMainImg" alt="主图"/>
        <button className="imgpost" onClick=
{()=>document.querySelector('#imgfile').click()} >点此插入新闻主图  插入后在左侧预览 仅在
首页显示</button>
        <input type="text" className="publishInputTitle" placeholder="新闻标题"/>
        <div   className="registerInfo" style={{textAlign:'center'}}></div >
        {/* E */}
        <div className="Editor" >
          {/* 将生成编辑器 */}
          <div ref="editorElem" style={{textAlign: 'left',height:'340px'}}>
          </div>
          <select className="selectType">
            <option>选择新闻主题</option>
```

```
            <option>科技</option>
            <option>娱乐</option>
            <option>政治</option>
            <option>游戏</option>
            <option>时尚</option>
            <option>军事</option>
            <option>其他</option>
          </select>
          <button className="publish" onClick={this.clickHandle.bind(this)}>发表新闻</button>
        </div>
      </div>
    )
  }
}
export default Publish;
```

第四章 结论

4.1 全文总结

本论文从原理到实现讲述了一个基于React, express , mongodb 的WEb新闻发布系统,虽然最终成功完成,但是仍然存在一些性能问题,并没有做React的性能优化,也没有时间写测试代码.没有由于时间关系,css只是粗略一些,而且没有考虑兼容问题,大量使用了ES6特性.而且最终的功能尚未完善,并没有完成初始设想的分类显示新闻功能,以及由于修改新闻主图会导致页面刷新的bug,最终无奈删除了修改新闻主图的功能.

4.2 后续工作展望

希望后续完成以下几个部分:

√ 增加修改主图,分类显示新闻功能

√ 优化网站速度

√ 优化兼容性,增强页面美观度

√ 引入 Redux/Mbox 管理数据

致谢

本项目的工作是在我的导师赵勇老师指导下完成的，感谢老师和学院鼓励我参与课程项目的设计，在指导老师的邮件指导和自己孜孜不倦的探索下，对于现代化前端开发有了更深的认识，对与接下来的本科学习有了更广阔的视野，同时培养了自己的自学能力和探索能力，在此谨代表个人向指导老师赵勇老师和学院致以最诚挚的感谢。

参考文献

[1] - node.js中文官网 (http://nodejs.cn/)

[2] - wxyyxc1992博客 (https://github.com/wxyyxc1992/Web-Series)

[3] - 维基百科 NoSQL (https://zh.wikipedia.org/wiki/NoSQL)

[4] - 深入浅出React和Redux / 程墨编著. -北京:机械工业出版社,2017.4