**附件4：**

河北师范大学本科生毕业论文（设计）模板

毕业论文（设计）册应包括如下内容：

1.任务书；

2.开题报告书；

3.文献综述；

4.外文文献及其译稿；

5.毕业论文（设计）正文：封面、声明页、摘要（中英文）、目录、正文及相关图表、注释、参考文献、附录（可选）、致谢等；

毕业论文全部文档应顺序装订成册，鼓励学生制作相应的电子文档，页面用A4纸。

未命名

**本科生毕业论文（设计）册**

学 院： 软件学院

专 业： H5

年 级： 2016

学生姓名： 白凯发

指导教师： 王勇

**河北师范大学本科生毕业论文（设计）任务书**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **姓 名** | **白凯发** | **学院** | **软件学院** | | **专业** | **H5** | **年级（班）** | **2016-8** |
| **毕业论文（设计）题目** | | 游戏玩家在线交流平台设计实现 | | | | | | |
| **指导教师** | **王勇** | **职称** |  | **教研室** | |  | **研究方向** |  |
| **论文（设计）基本要求：**完成游戏玩家在线交流平台的设计与开发，并能对用户的操作进行管理，在答辩时能够对设计的游戏玩家在线交流平台系统进行演示。 | | | | | | | | |
| **论文（设计）研究目标：**  利用当下流行的mobx Vue nodejs框架,并运用github进行项目管理，实现基于web的游戏玩家在线交流平台，旨在完成一个交互性强、内容丰富的玩家的交流平台。  (1) 实现设计内容的全部功能；  (2) 大量查阅官方文档，熟悉设计内容，查询设计方法；  (3) 查阅并翻译一篇与设计相关的外文资料；  (4) 开发出一整套完整的项目系统；  (5) 按照论文撰写格式完成毕业论文，并参加论文答辩；  (6) 论文答辩进行系统运行演示。 | | | | | | | | |
| **主要参考文献：**  [1] <https://cn.mobx.js.org/> mobx官方中文文档.  [2]https://www.baidu.com/link?url=Fh5y1h00OYUkk-X6NjYFBywduiKyBZssrpL1Q\_IJGw\_kc-SgIrCWh9GmLBkYp5Fo&wd=&eqid=8d54df07000a7dc4000000035e5ba15b mongodb官方文档.  [3]http://nodejs.cn/ nodejs官方中文文档 | | | | | | | | |

指 导 教师： 年 月 日

教研室主任： 年 月 日

**河北师范大学本科生毕业论文（设计）开题报告书**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **姓 名** | | **白凯发** | **学院** | **软件学院** | **专业** | **H5** | **年级（班）** | **2016-8** |
| **论文题目** | | 游戏玩家在线交流平台设计实现 | | | | | **指导教师** | **王勇** |
| **课题论证** | 当前社会人们的生活丰富多彩，网络游戏又极大的丰富了人们的生活内容，不同类型的游戏带给了人们不同的感受。例如斗地主，麻将等等休闲类的游戏给人们在繁忙的工作之余一丝休息的片刻，调整大脑状态，更好的投入工作生活当中，游戏对于人们的生活起着越来越重要的作用，但是游戏想要运行出色，除了要有良好的优美的制作画面，丰富引人的内容外，还需要玩家们能够多多交流，保持一定的话题热度，才能更好的促进游戏发展，所以探求快速开发网站的技术。 | | | | | | | |
| **方案设计** | 本课题将从交流平台的内容上来论证上述观点，游戏玩家交流平台主要是方便玩家们交流在游戏中的心得，以及反馈游戏中存在的问题，游戏玩家交流平台至少要包含用户的个人信息，以及发帖登基础功能。准备通过设计前端页面和后台数据管理来实现这一可视化操作。 | | | | | | | |
| **进度计划** | 第一阶段（第1周）：查询资料，熟悉设计内容，掌握设计方法；  第二阶段（第2周）：撰写开题报告；  第三阶段（第3周）：初步进行系统分析、需求分析；  第四阶段（第4－6周）：设计数据库模型，搭建Mobx与Nodejs框架；  第五阶段（第7—10周）：界面开发，系统开发；  第六阶段（第11—12周）：整理文档，撰写论文；  第七阶段（第13周）：毕业答辩。 | | | | | | | |
| **指**  **导**  **教**  **师**  **意见** | **指导教师：**  **年 月 日** | | | | | | | |
| **教研室意见** | **教研室主任：**  **年 月 日** | | | | | | | |

**河北师范大学本科生毕业论文（设计）文献综述**

|  |
| --- |
| 从总体上来说，vue,react,angular三分天下，在国外主要以angular和react为主，国内主要以react和vue为主，三大框架很多思想都具有相似性，其中包括   1. 都使用了 Virtual DOM，虚拟DOM相比传统直接操作DOM提升了性能 2. 组件化，每一个小的内容都是一个组件，写代码像堆乐高积木，非常灵活 3. react和vue都有props的概念，都可以通过props来进行父子组件传值， 4. react和vue都有自己的构建工具，状态管理方法 5. react和vue都有很好的chrome扩展工具，扩展工具能够协助我们找出bug ，让我们看到vue或者react中的数据变化 6. react和vue都有相应的配套框架，例如 ui框架，react配合antd，vue配合elementui，状态管理办法，各自的路由等等   不同之处在于   1. vue的做法在于是鼓励写常见的HTML模板，写起来的样子接近标准的HTML，只不过是多了一些属性而已；react则是推荐你完全写JSX语法；但是需要注意的是vue在技术上其实也是支持Render函数和JSX，只是不是默认的而已 2. vue中state并不是必须的，数据则是由data属性在vue的对象中独立进行管理，data其实就是应用中数据的保存者，你可以直接进行修改；react中的state在应用中不仅仅是不可变的，它不能被直接修改，但是可以通过SetState方法来进行更新 3. RN即react native可以在手机上创建原生的应用，目前处于所有手机应用的领先位置；weex仍处于开发状态，目前还没有经过实际项目的验证 4. vue的体积更小，渲染速度更快，性能更好，语法更简单，并且上手更快，是一个渐进式的框架，它专注于视图层；但是同样的react性能也很好 5. vue框架应用广泛，是目前很火的框架，并且关注的人很多，但是react社区庞大，并且是目前非常流行的框架，它的背后有FaceBook撑腰，发布的时间比vue更早，并且在生产方面经过很好的测试；其次在GitHub上，vue的start数已经超过了react   我预测在国内，仍会以vue与react为主，不过未来vue的发展会越来越好，因为其上手相对容易，更容易吸引大量的初学者。但是无论是vue与react，都需要配合一定的其他框架来扩展能力，而这类的框架相对复杂，在对几种框架进行分析后，选定mobx作为配合框架，配合react来快速开发迭代项目。文献资料中，nodejs是一切的基础，所有的包依赖都需要nodejs，mongodb让我们有存储数据的能力，mobx使得我们能快速开发项目。 |

**河北师范大学本科生毕业论文（设计）翻译文章**

|  |
| --- |
| Mobx的简单使用  1. 需要使用@observable 定义状态并使其可观察  你可以用任何你喜欢的数据结构来存储状态，例如数组、类、对象。引用、 循环数据结构等等，这些都没有关系。 只需要确保所有改变的属性打上 mobx 的标记使它们变得可观察即可。  import {observable} from 'mobx';  var appState = observable({  timer: 0  });  2. 创建视图用以响应数据状态的变化  我们的 appState 目前还没有观察到任何的东西。 此时你可以创建视图，当 appState 中相关数据发生改变时视图便会自动更新。 MobX 就会以一种最小限度的方式来更新视图。 事实上这一点帮助我们节省了大量的样板文件，不仅如此它有着令人匪夷所思的高效。  一版来说，所有的函数都可以成为可以观察自身数据的响应式视图，MobX这一框架 可以在任何符合ES5的JavaScript环境中去应用。 但是在这里所用的示例是 却是ES6版本的 React 组件视图。  import {observer} from 'mobx-react';  @observer//装饰器语法  class TimerView extends React.Component {  render() {  return (<button onClick={this.onReset.bind(this)}>  Seconds passed: {this.props.appState.timer}  </button>);  }  onReset () {  this.props.appState.resetTimer();  }  };  ReactDOM.render(<TimerView appState={appState} />, document.body);  3. 更改状态  我们第三件要做的事就是更改状态。 也就是你的应用到底要做什么。 不同于一些其它框架，MobX 不会去命令你如何如何去做，这是一点很大的不同，意味着给你的自由度变高，同时省略了一些繁琐的固定格式的内容， 这是最佳的实践，但我们务必关键要记住一点: MobX 帮助我们以一种简单直观的方式来完成我们的工作。  一个样例代码，下面的代码每秒都会修改我们的数据，而当需要的时候UI会自动更新（非常神奇）。 无论此时是在改变状态的控制器函数中，还是在应该更新的视图函数中，这里都没有明确的关系定义。 使用 observable 来装饰我们的状态和视图，这就足以让 MobX检测所有的关系了。  appState.resetTimer = action(function reset() {  appState.timer = 0;  });  setInterval(action(function tick() {  appState.timer += 1;  }), 1000);  其实只有在严格模式(默认是不启用)下使用 MobX 时才需要 action 包装。 建议使用 action，因为它将帮助我们更好地组织应用，并且能够表达出一个函数修改状态的意图。 与此同时,它还自动应用事务以获得最佳性能。  --<https://cloud.tencent.com/developer/section/1489464>  Mobx仅仅需要三步即可完成对数据的存储使用 |

未命名

**本科生毕业论文（设计）**

题目：游戏玩家在线交流平台设计实现

学生姓名： 白凯发

指导教师： 王勇

学 院： 软件学院

专 业： H5

年 级： 2016

完成日期： 2020 年 4 月 15 日

学位论文原创性声明

本人所提交的学位论文 游戏玩家在线交流平台设计实现，是在导师的指导下，独立进行研究工作所取得的原创性成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中标明。

本声明的法律后果由本人承担。

论文作者（签名）： 白凯发 指导教师确认（签名）：

2020年 4 月 15 日 年 月 日

学位论文版权使用授权书

本学位论文作者完全了解河北师范大学有权保留并向国家有关部门或机构送交学位论文的复印件和磁盘，允许论文被查阅和借阅。本人授权河北师范大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其它复制手段保存、汇编学位论文。

已申请保密的论文须添加本句。

保密的学位论文在\_\_\_\_\_年解密后适用本授权书

论文作者（签名）： 白凯发 指导教师（签名）：

2020年 4 月 15 日 年 月 日

摘 要

在对游戏玩家在线交流平台设计实现快速搭建技术的探究过程中，为了达到快速开发的目的，探究了前端三大框架，主要是vue与reacat在**开发效率**上的异同，以及后端关于**nodejs**应当使用什么相关技术的一些辩论。主要采用了对比的手段进行论证，通过比较各自的优缺点来进行选择论证，最终通过比较选择了**react**框架配合的数据存储框架是**mobx**，后端选择nodejs以缩小开发的技术成本。最终实现了一个针对游戏玩家的在线交流平台，通过这次实践发现技术选型非常决定开发的效率，这在企业级开发中非常重要，**高效**的开发能够使我们尽可能快的完成任务，抢占市场先机。

**Abstract**

In the process of exploring the design and implementation of online communication platform for gamers, in order to achieve the purpose of rapid development, the three front-end frameworks are explored, mainly the similarities and differences between Vue and reacat in **development efficiency**, as well as some debates on what technologies should be used by **nodejs** in the back-end. It mainly uses the comparative method to demonstrate, through comparing their advantages and disadvantages to choose the demonstration, and finally through the comparison, it selects the data storage framework that the **react** framework cooperates with is **mobx**, and the backend selects nodejs to reduce the technical cost of development. Finally, an online communication platform for game players is realized. Through this practice, it is found that technology selection determines the efficiency of development, which is very important in enterprise level development. **Efficient development** can enable us to complete the task as soon as possible and seize the market opportunity.

目录

[第1章 后台技术选型与实现](#_Toc12763_WPSOffice_Level1) [1](#_Toc12763_WPSOffice_Level1)

[1.1技术选型：](#_Toc12763_WPSOffice_Level2) [1](#_Toc12763_WPSOffice_Level2)

[1.2 着手后端开发](#_Toc11044_WPSOffice_Level2) [3](#_Toc11044_WPSOffice_Level2)

[1.3后台开发细节补充](#_Toc24415_WPSOffice_Level2) [4](#_Toc24415_WPSOffice_Level2)

[第2章 前台技术选型与实现](#_Toc11044_WPSOffice_Level1) [6](#_Toc11044_WPSOffice_Level1)

[1.1技术选型](#_Toc15642_WPSOffice_Level2) [6](#_Toc15642_WPSOffice_Level2)

[1.2前端开发](#_Toc25751_WPSOffice_Level2) [9](#_Toc25751_WPSOffice_Level2)

[参考文献 1](#_Toc15642_WPSOffice_Level1)0

[附 录 1](#_Toc25751_WPSOffice_Level1)1

[致 谢 1](#_Toc23619_WPSOffice_Level1)2

游戏玩家在线交流平台设计实现

1. 后台技术选型与实现

1.1技术选型：

由于之前在老师的帮助下，已经进行过实战项目的开发，因此毕业设计课题定为游戏玩家在线交流平台设计实现,后台，我则是选择了nodejs，选择nodejs的原因主要是Nodejs区别于传统应用1. 它是异步事件驱动2.它是非堵塞的I/O，3.它是单线程（这里指主线程）4.它的性能出众，我对以上4点进行详细的阐述：

一、首先事件驱动，所谓的事件驱动，就是指在持续事务管理的过程中，进行的一种决策策略，即可以跟随当前时间点上出现的事件，灵活的调动可以调用的资源，执行与之相关的任务，使得不断的出现的问题得以快速解决，防止事务堆积造成堵塞。其次Nodejs的设计思想是以事件驱动为核心，事件驱动的核心在于异步回调，它提供的大多数api基本上都是基于事件的、是种异步的风格。而事件驱动的优势就在其能够充分的利用系统资源，执行代码无须阻塞等待某种操作的完成，有限的资源可以用于去做一些其他的任务。事件驱动机制是通过内部单线程高效率地维护事件循环队列来实现的，没有多线程的资源占用和上下文的切换。

二、异步、非堵塞的I/O，Nodejs提供的很多模块中其实都是异步执行的。例如，一个文件操作函数。一个异步I/O大致流程是这样的：1. 主模块发起I/O调用，①用户可以通过js代码调用nodejs的相关核心模块，此时将回调函数和参数传入这个核心模块，②再将回调函数与参数封装2.开始执行回调，执行回调主要由两步完成①操作完成将结果的时候储存到请求对象的result属性上，此时会发出完成通知。②循环事件，如果此时仍有未完成的，那么就在进入对象请求的I/O观察者队列之中，在此之后会当做事件处理；

三、单线程

Nodejs其实跟Nginx一样都是以单线程为基础的，这里的单线程主要是指主线程为单线程，所有的阻塞的进程全部放入一个线程池中，然后主线程通过队列的方式跟线程池来进行相互协作。我们写js的部分其实不需要关心线程的问题，只需要简单了解就可以了，主要由一堆callback回调构成的，方便我们使用，然后主线程在循环过在适当场合调用即可。

四、性能出众

底层选择是用c++和v8来实现的，在上面第一点曾经讲到过，nodejs的事件驱动机制，这就意味着面对大量的http请求，nodejs是完全是通过事件驱动来完成任务的，性能这部分完全不用担心，并且非常的出色。

Nodejs的适用场景：

比较适合I/O密集型的应用，例如多人在线小游戏，**在线多人聊天**，实时新闻，博客，微博等等之类的。

不适合的场景有以下几种：cpu密集型的应用，比如计算圆周率，视频解码等业务场景较多的。

我们的项目属于在线多人聊天，自然选择nodejs有着得天独厚的优势

我是如何判断**在线多人聊天是**I/O密集型的呢，下面详细介绍下CPU密集型和I/O密集型的区别：

CPU密集型（CPU-bound）

CPU密集型同时也叫做计算密集型，它指的是我们的系统的硬盘、内存性能、相对CPU要好很多很多，此时，系统在运作的时候大部分的状况是CPU Loading 100%，CPU要读/写I/O(硬盘/内存)，I/O在很短的时间完成，但CPU仍然有许多运算需要处理，CPU 的Loading很高。

在多重程序的系统中，大部份的时间用来做计算和逻辑判断等CPU动作的程序被称为CPU bound。比如说一个计算圆周率到小数点一千位以下的program，在执行的过程当中绝大部份时间其实用在三角函数和开根号的计算，这种程序属于CPU bound程序。

CPU bound的程序一般来说CPU的占用率相当高。这可能是因为任务不常常访问I/O设备，也有可能是因为程序是多线程的实现因此实际上屏蔽掉了等待I/O的时间。

IO密集型（I/O bound）

IO密集型指的是系统的CPU性能相对硬盘、内存要好很多很多，这种情况下，系统运作，大部分的状况是CPU在等I/O (硬盘/内存) 的读/写操作，此时CPU Loading并不太高。

I/O bound的程序一般情况下在达到性能极限的时候，CPU占用率仍然处于较低的状态。这很有可能是因为任务本身需要大量的I/O操作，然而pipeline做得不是很好，并没有充分利用处理器能力。

CPU密集型 vs IO密集型

我们可以把任务分为计算密集型和IO密集型。

计算密集型任务的特点主要是需要进行大量的计算，这时会进行CPU资源的消耗，例如圆周率的计算、视频的高清解码等等，依靠着CPU的运算能力。这种计算密集型任务虽然可以用使用多任务完成，但是问题在于任务越多，花在任务切换的时间也就越多，CPU执行任务的效率就越来越低，因此，我们想要最高效地利用CPU，计算密集型任务的同时要进行的数量应当等于CPU的核心数。

计算密集型任务因为主要使用CPU资源，因此，代码运行的效率非常重要。Python这种脚本语言的运行效率很低，并不适合做计算密集型的任务。对于计算密集型任务，用C语言编写更好。

第二种任务的类型是IO密集型任务，主要包括网络、磁盘IO的任务都是IO密集型任务，这类任务的特点是CPU消耗不多，任务大部分时间在等待IO操作的完成（因为IO的速度低于CPU和内存的速度非常多）。对于IO密集型的任务，任务量越多，CPU的运行效率越高，但也不是越高越好，也需要有一个限度。常见的大部分任务都是IO密集型任务，例如我们的Web应用。

IO密集型在任务执行期间，99%的时间都花在了IO上，花在CPU上的时间相对来说很少，因此，用运行速度极快的C语言来替换Python这样运行速度极低的脚本语言，完全无法提升运行的效率。对于IO密集型任务，最合适的语言就是开发效率最高（代码量最少）的语言，那么肯定的脚本语言是首选，C语言自然是最差的选择。

数据库选择使用方便的mongodb，主要基于mongodb有以下优点：

1、弱一致性（最终结果是一致的），更容易保证用户的访问速度

2、文档结构的存储方式，使得我们能够更便捷的获取所需要的数据

比如对于一个层级式的数据结构来说，如果要将很多的数据采用扁平式的或表状的结构来保存数据，这不管是在查询还是获取数据时都会异常困难。

3、第三方的支持非常多。

与NoSQL相比，MongoDB具有的优势，主要源于现在网络上的很多NoSQL开源数据库完全属于社区型的，并没有官方支持，给使用者带来了很大的不可预知的风险。

然而开源的MongoDB文档数据库背后有商业公司为它提供商业培训和支持。

4、性能非常优越

在某些使用场合下，例如千万级别的文档对象，有将近10个G的数据，对有索引的ID的查询并不比mysql慢，并且对非索引字段的查询，那便是全面胜出。 mysql实际上并不能胜任大数据量下任意字段的查询，但是mongodb的查询性能可以，同时它的写入性能也非常厉害，可以写入百万级别的数据。

1.2 着手后端开发

后端开发，首先是架构设计，既然选择了express框架，那就要配合mongoose，但是如何解析客户端请求的body中的内容呢？我想到了bodyparser，它用于解析客户端请求的body中的内容,内部使用JSON编码处理,url编码处理以及对于文件的上传处理

。但是当我们登录的时候，如何验证我们的用户名与密码呢？这次我才用的是passport，passport是Nodejs的一个中间件，主要用于用户名和密码的登陆验证。在项目中我常常用它来验证后台用户名和密码，但passport其实更多用在第三方登录，功能非常强大举例一个passport的使用。

app.get('/api/users/me',

passport.authenticate('basic', { session: false }),

function(req, res) {

res.json({ id: req.user.id, username: req.user.username });

});

除此之外，我还需要对后端项目进行路由的规划，所谓路由，就是如何处理HTTP请求中的路径部分。举个例子“[http://xxx.com/users/m](http://xxx.com/users/profile)e”这个URL，路由将决定怎么处理/users/me这个路径，最直接的路由配置方法，就是我们调用app.get()、app.post()一条一条的配置，不过对于大型项目来讲，这会搞出人命来的。所以呢，我们实际开发中需要结合路由参数（query string、正则表达式、自定义的参数、post参数）来减小工作量提高可维护性。

除此之外，还需要对上传静态资源文件等等进行规划，于是我是用了multer中间件。

Multer中间件是Express官方推出的，是用于Node.jsmultipart/form-data请求数据处理的中间件。

它能够高效的处理文件上传，但并不处理multipart/form-data之外的用户请求。

1.3后台开发细节补充

上一小节主要介绍我大致使用了那些比较重要的重要节点的技术细节，用来解决大方向上的，一些服务的基础功能，然而对于后端开发，仅仅只有这些基础服务还不够，为了我们网站的安全性，我们需要引入一下几点内容。其中，

1.3.1被抛弃的cookie

cookie的作用主要是我们发送post请求到后端，后端验证合法，返回响应，并Set-Cookie为sessionid=\*\*\*;username=\*\*\*，浏览器接到响应发Set-Cookie,将其存入内存中；浏览器再次发起请求会带上Cookie信息sessionid=\*\*\*;username=\*\*\*，服务器根据sessionid验证当前用户，根据username,查找对应数据，这是一个cookie设置与利用过程。但是Cookie容易被篡改呢，cookie是存储在客户端的，用户可以任意修改cookie值，比如如果当前用户知道username的作用，修改username,根据username,查找数据库中的对应数据，这样数据有被恶意篡改的风险。我们就要给cookie增加签名，如服务器接收到username=fire||34sdklkas,然后使用签名生成算法secret(fire)=666,得到的签名666和请求中数据的签名不一，则证明数据被篡改，不予通过，所以cookie中不应该存储敏感数据。

1.3.2 jwt-token的使用

由于http协议是无状态的，客户端每次访问都是新的请求。每次请求都需要进行身份验证，传统方式是用session+cookie上面介绍过，而JWT就是更安全方便的方式。它的特点简洁，紧凑，不占空间，传输速度快,有利于多端分离，接口的交互，JWT是一种Token规范，主要面向的还是登录、验证和授权方向，同时可以用只来传递信息。

之所以选择jwt-token的原因主要有以下几点

(1)相比于 session jwt-token少了读取文件的步骤，效率高

(2)天生免疫 CSRF 攻击

(3)不用担心用户禁用 cookie ；不用悬挂需要使用 cookie 的提示信息；

1. 前台技术选型与实现

1.1技术选型

1.1.1三大框架选取

首先前端三框架中我选用的是react。

它的优点主要有

1、React速度很快：不直接操作DOM，而是引入了虚拟DOM，性能非常好。

2、跨浏览器兼容：虚拟DOM帮助我们解决了跨浏览器的问题，提供了标准化的API，以至于在IE8中都可以正常运行。

3、一切都是组件：模块化的代码，重用代码容易，可维护性比较高。

4、单向的数据流：Flux是一个用于在JavaScript单向数据层的[架构](http://lib.csdn.net/base/16" \o "大型网站架构知识库" \t "https://www.cnblogs.com/cxying93/p/_blank)，随着React视图库的开发进而被Facebook概念化。

**1.1.2数据存储框架对比(mobx与redux)**

**函数式和面向对象**

​ Redux更多的是遵循函数式编程的思想，而Mobx则更多从面相对象角度来考虑问题。

​ Redux提倡的是编写函数式代码，

​ Mobx设计则更多偏向于面向对象编程和响应式编程，通常是将状态包装成可观察的对象，于是我们拥有了可观察对象的所有能力，一旦状态对象变更，就能自动获得更新。

**单一store和多store**

​ store是应用管理数据的位置，在Redux应用中，我们习惯于将所有共享的应用数据集中在一个大的store中，而Mobx则通常按模块将应用状态划分，在多个独立的store中管理。这样做的好处就是使用非常方便，随用随存随取。

**JavaScript对象和可观察对象**

​ Redux默认以JavaScript原生对象的形式存储数据，而Mobx则使用可观察对象：

​ Redux需要我们手动追踪所有状态对象的变更，这需要你自动更新变化的值，实际开发中每次都需要变化会让你感觉非常不爽，效率极低。

​ Mobx中可以监听可观察对象，当其变更时将自动触发监听。使用起来非常舒服。

**不可变（Immutable）和可变（Mutable）**

​ Redux状态对象通常是不可变的（Immutable）：

​ 这意味着我们不能直接操作状态对象，而是在原来状态对象基础上返回一个新的状态对象，这样能很方便的使用应用上一个状态；而Mobx中则可以直接使用新值更新状态对象。

**mobx-react和react-redux**

使用Redux和React应用连接时，需要使用react-redux提供的Provider和connect：

1. Provider：将Store注入React应用；
2. connect：将store state注入容器组件，并选择特定状态作为容器组件props传递；

对于Mobx而言，同样需要两个步骤：

1. Provider：使用mobx-reac提供的Provider将所有stores注入应用；
2. 使用injec将特定store注入某组件，store可以传递数据；使用observer保证组件能响应store中的可观察对象（observable）变更。

1.1.3 为什么使用hooks

## 优点

### 一、更容易复用代码

通过自定义hooks来复用状态，解决了类组件难以复用逻辑的问题。类组件的逻辑复用方式是“高阶组件”和renderProps。然而hooks是这样解决的

1. 每调用useHook一次都会生成一份独立的状态，函数每次调用都会开辟一份新的独立的内存空间。
2. 虽然状态和副作用的存在依赖于组件，但它们可以在组件外部进行定义。这点是类组件做不到的，你无法在组件外部声明state和副作用

基于上面两点，高阶组件和renderProps同样可以。但hooks有以下几点优势：

1. 结构更简单，不用层层嵌套。相对比高阶组件“祖父=>父=>子”的结构，hooks是这样的： const { brother1 } = usehook1; const { brother2} = usehook2; 把复用的行为放在业务代码的同一层级，结构清晰容易理解。
2. 写一个custom hook对比高阶组件的代码量少了很多，人们更愿意随时随地复用代码。

### 二、保持清爽的代码风格

函数式组件，、函数式编程风格状态，在运行环境、每个功能都包裹在函数中，风格清爽优雅。

### 三、代码量少

1. 向props或状态取值更加方便，函数组件的取值都从当前作用域直接获取变量，而类组件需要先访问实例引用this，再访问其属性或者方法，多了一步。

四、更容易发现无用的状态和函数

对比类组件，函数组件里面的unused状态和函数更容易被发现

### 五、更容易拆分组件

写函数组件的时候，你会更愿意去拆分组件，因为函数组件写起小组件比类组件要简单。

1.2前端开发

1.2.1技术框架搭建

想要写好一个项目，一个好的框架必然不可少，为了使用上react hooks，因此采用mobx-react-lite来兼容之前的书写格式。这里描述下具体的兼容问题在哪里，我没有采用使用上下文的方式来引入mobx,因为上下文的方式首先增加了更多的api，其次，上下文的这种方式并不便于理解与掌握整体的架构，再者上下文的方式使得我们回到了过去，仓库之前没有分离开来，非常不便与我们对某些文件下某些具体数据的操作，冗余度高。因此采用mobx-react-lite是至关重要的，mobx-react-lite是通过useContext，createContext，useStores，useObserver这些api来保证我们在组件中可以方便的使用，对比之前的@inject('')与@observer,完美的配合在了一起，无论想使用类组件亦或者是函数组件，都能轻松自如，游刃有余的使用我们的公有数据。

1.2.2项目依赖

项目依赖antd和装饰器语法，因此采用customize-cra工具来简化这一操作。

customize-cra的优缺点主要有以下几点

优点:1.不需要npm run eject打开webpack配置来进行操作，减少了配置时间

1. 一次配置，多次使用，每次更换项目只需要移动config-overrides.js文件即可，方便快捷

缺点:主要缺点就是不打开webpack配置，无法进行一些精细化的操作，有些配置不存在于customize-cra中，当我们需要对webpack进行精细化配置的时候，customize-cra就显得捉襟见肘了。

1.2.3前端组成部分

前端主要有个人中心，登录注册，热门咨询，热点轮播，以及进入其内部后的讨论组成。

注册：

登录注册要考虑的是用户名密码存储在哪里，那么肯定是存在数据库中，但是如果直接存储在数据库中是非常不安全的，历史上数据库被黑掉以及拖走数据库的案例数不胜数，因此我需要保证即使数据库被黑掉，被拖走其它人也不能从数据库中得到用户的信息。因此我考虑了bcrypt加密算法之所以没有选择之前使用过的md5算法，主要是考虑到一下因素。

 BCrypt，是一个跨平台的文件加密工具。BCrypt强哈希方法，每次加密结果都不一样。这就对安全性而言具有强安全性；但其也有缺点，性能较差，bcrypt是牺牲了性能来保全它的安全性的。

登录:为什么选择token，最简单的token组成:uid、time、sign组成，我们还可以把不变的参数也放进token，避免多次查库，对比其它几种方式，作为身份认证 token安全性比session好，因为每个请求都有签名并且还能防止监听以及重放攻击，而session就必须靠链路层来保障通讯安全了。如上所说，如果你需要实现有状态的会话，仍然可以增加session来在服务器端保存一些状态。

参考文献

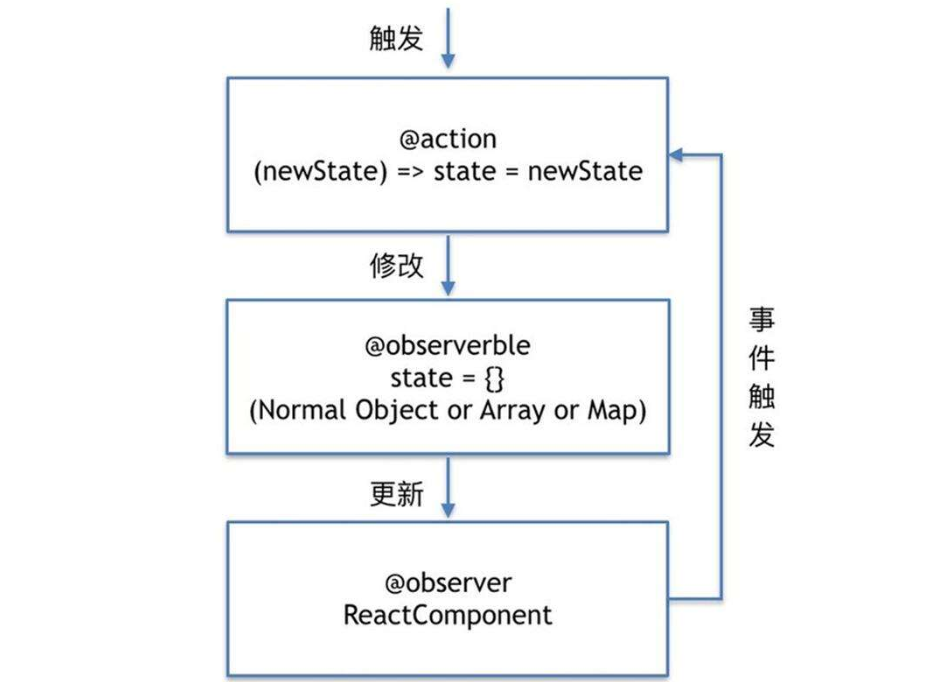
[1] <https://cn.mobx.js.org/> mobx官方中文文档.

[2]https://www.baidu.com/link?url=Fh5y1h00OYUkk-X6NjYFBywduiKyBZssrpL1Q\_IJGw\_kc-SgIrCWh9GmLBkYp5Fo&wd=&eqid=8d54df07000a7dc4000000035e5ba15b mongodb官方文档.

[3]http://nodejs.cn/ nodejs官方中文文档

附 录

mobx工作流程



致 谢

历史两个月终于完成了这篇论文，在这个充满艰辛的过程中，从过去的一脸懵逼什么是论文到逐渐意识到论文是一样什么样的东西，再到想尽各种办法查阅资料去翻看应该怎么写论文，论文有哪些内容，需要注意哪些格式，在此感谢本论文所引用的各位学者的著作，也要感谢国内良好的社区环境提供了大量的资料支持，同时感谢指导老师的谆谆教诲，督促我及时高效的完成任务，由于我的学术水平有限，所写论文难免有不足之处，恳请各位老师和同学批评指正，感谢。