未命名

**本科生毕业论文（设计）册**

学 院： 软件学院

专 业： H5

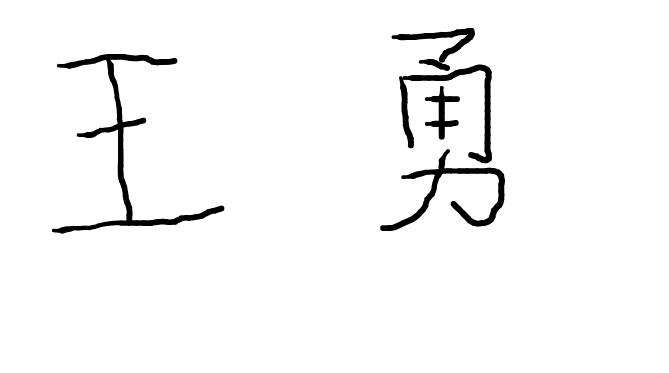
年 级： 2016

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** | 工程师 | **教研室** | | H5教研室 | **研究方向** | Web架构 |
| **论文（设计）基本要求：**利用当下流行的mobx react 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。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：**  (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/lrCWh9GmLBkYp5Fo&wd=&eqid=8d54df07000a7dc4000000035e5ba15b mongodb官方文档.  [3]http://nodejs.cn/ nodejs官方中文文档 | | | | | | | | |



指 导 教师： 2019 年 11 月 15 日

教研室主任：  2019 年 11 月 15 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，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，游戏对于人们的生活起着越来越重要的作用，需要玩家们能够多多交流，保持一定的话题热度，所以探求快速开发网站的技术，以及一个玩家交流平台应该是怎么样的才能促进玩家之前的联系，增强游戏的热度与趣味性。 | | | | | | | |
| **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登录注册，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来实现这一可视化操作，方便用户使用。 | | | | | | | |
| **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；界面开发，系统开发  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11—12周）：整理文档，撰写论文，毕业答辩； | | | | | | | |
| **指**  **导**  **教**  **师**  **意见** | **指导教师：**  **2019年 12月 1 日** | | | | | | | |
| **教研室意见** | **教研室主任：**  **2019年 12月 1 日** | | | | | | | |

**河北师范大学本科生毕业论文（设计）文献综述**

|  |
| --- |
| 一:前言  交流平台对于任何一个游戏项目来说都有着举足轻重的地位，我们不可能再回到那个各自为战，孤零零娱乐的时代，如今国内外电子竞技行业火热，甚至在前些年已经被纳入奥运会项目，可以见得之所以电子竞技能够这么火热，是因为有很多人在关注这件事情，那么游戏的交流平台在其中就起到了至关重要的作用，那么一个玩家交流平台要有那些东西呢？怎么样的交流平台对于用户来说是体验良好的呢？从开发人员的角度上，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，三大框架应该如何抉择呢？  二:主题  本论文旨在通过对现有的已经存在的平台，经过分析演练，以及网络上的一些对平台优劣的讨论，进行有选择的尝试采用不同的技术手段，来实现我们的交流平台，在这个过程中，会根据技术的特点，有选择性的取用，根据交流平台功能，有选择的进行选取，最终实现一个能够方便玩家们交流平台。  三:总结  总的来说，这次游戏交流平台撰写的前端功能主要有用户的登录注册，优美的视频背景，用户个人中心，帖子查看，评论等功能，后端的功能主要有jwt-token校验，用户密码加密，数据获取等等，通过这些功能，用户便能够通过这样的一个平台来实现。 |

**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>  原文：  Simple use of mobx  1. You need to use @ observable to define the state and make it observable  You can use any data structure you like to store States, such as arrays, classes, and objects. References, circular data structures, and so on are irrelevant. Just make sure that all changed attributes are mobx marked to make them observable.  import {observable} from 'mobx';  var appState = observable({  timer: 0  };  2. Create views to respond to changes in data status  Our appstate hasn't seen anything yet. At this time, you can create a view, which will be updated automatically when the relevant data in appstate changes. Mobx updates the view in a minimal way. In fact, this has helped us save a lot of template files, not only that it has an amazing efficiency.  In the first version, all functions can be a responsive view of their own data. Mobx framework can be applied in any JavaScript environment conforming to Es5. But the example used here is the react component view of ES6.  import {observer} from 'mobx-react';  @Observer / / decorator syntax  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. Change status  The third thing we have to do is change the state. That's what your app does. Unlike some other frameworks, mobx does not command you how to do it. This is a big difference, which means that it gives you more freedom and omits some tedious fixed format content. This is the best practice, but we must remember one key point: mobx helps us complete our work in a simple and intuitive way.  For a sample code, the following code will modify our data every second, and the UI will update automatically when necessary (it's amazing). No matter in the controller function of changing state or in the view function that should be updated, there is no clear relationship definition here. Using observable to decorate our state and view is enough for mobx to detect all relationships.  appState.resetTimer = action(function reset() {  appState.timer = 0;  };  setInterval(action(function tick() {  appState.timer += 1;  }), 1000);  In fact, the action wrapper is only needed when mobx is used in strict mode (not enabled by default). Action is recommended because it will help us better organize the application and express the intention of a function to modify the state. At the same time, it automatically applies transactions for optimal performance. |

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

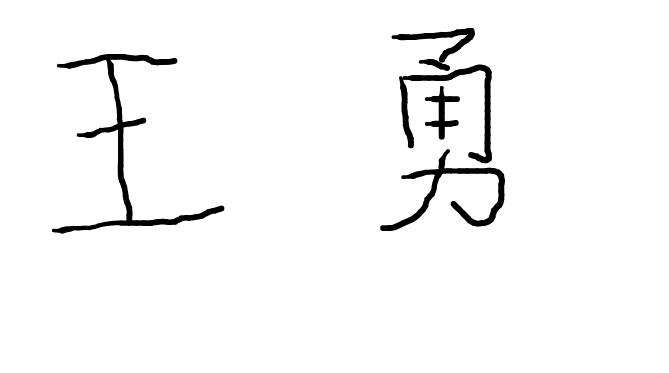
年 级： 2016

完成日期： 2020 年 5 月 13 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。

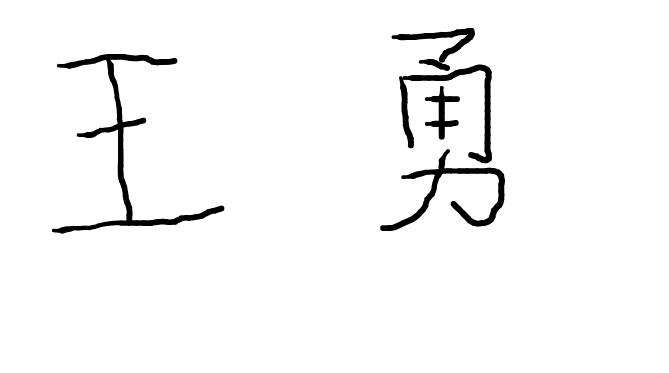


论文作者（签名）：  指导教师确认（签名）：

2020年 4月 15日 2020年 4月15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。



论文作者（签名）：  指导教师（签名）：

2020年 4月 15日 2020年 4月15日

摘 要

目的: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，为了达到快速高效的开发，以较低的技术成本，较高的效率以及为了达到能够方便玩家交流，方便官方发放游戏公告的目的。

方法:此次探究了前端两大大框架，通过对比论证的手法，来彰显出使用某种技术的优势，这会在主题部分进行详细阐述，在页面展示上，通过多种不同多样化的内容，丰富加深用户的使用感受。

结果与结论:通过参考其它交流网站，主要通过设置精美背景来增加网站的美观度，通过轮播，新闻等等方式丰富网站的内容，通过比对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。通过比较各自的优缺点最终选择了react框架，配合的数据存储框架是mobx，后端则是选择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。

**关键词： 开发效率，nodejs，react，mobx**

**Abstract**

Objective: in the process of exploring the design and implementation of the online communication platform for gamers, in order to achieve the purpose of rapid development and facilitate the exchange of gamers.

Methods: this time, we explored the two front-end frameworks, and demonstrated the advantages of using a certain technology by means of comparative argumentation. This will be elaborated in the theme part, and on the page display, we will enrich and deepen the user's use experience through diversified content.

Results & CONCLUSION: by referring to other communication websites, we mainly increase the beauty of the website by setting beautiful background, enrich the content of the website by means of rotation, news, etc., compare the similarities and differences between Vue and reacat in development efficiency, and some debates on what relevant technologies should be used by nodejs in the back end. By comparing their advantages and disadvantages, we choose react framework, mobx as the data storage framework, and nodejs as the backend to reduce the development cost. 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.

**Keywords:** **development efficiency,nodejs,react,mobx**

目 录

[第1章 绪论 1](#_Toc40775805)

[1.1 课题存在的背景以及意义 1](#_Toc40775806)

[1.2 市场存在的平台现状 1](#_Toc40775806)

[第3章 后端开发 4](#_Toc40775807)

[3.1 技术框架设计 4](#_Toc40775808)

[3.1.1 数据库设计 4](#_Toc40775809)

[3.1.2 静态资源文件存储 6](#_Toc40775810)

[3.1.3 后端路由规划 6](#_Toc40775811)

[3.1.4 前台内容解析 6](#_Toc40775812)

[3.1.5 实时通信 6](#_Toc40775813)

[3.1.6 用户校验（1） 6](#_Toc40775814)

[3.1.7用户校验（2） 6](#_Toc40775815)

[3.2技术选型的原因： 7](#_Toc40775816)

[第4章 前端UI设计 8](#_Toc40775817)

[第5章 前端程序开发 9](#_Toc40775818)

[5.1技术选型 9](#_Toc40775819)

[5.2前端开发 9](#_Toc40775820)

[5.2.1技术框架设计 9](#_Toc40775821)

[5.2.2技术选型的原因 10](#_Toc40775822)

[5.2.3注册模块 11](#_Toc40775823)

[5.2.4个人中心 12](#_Toc40775824)

[5.2.5热点轮播 12](#_Toc40775825)

[5.2.6热门咨询 14](#_Toc40775826)

[参考文献 15](#_Toc40775827)

[附 录 16](#_Toc40775828)

[致 谢 17](#_Toc40775829)

论文题目：游戏玩家在线交流平台设计实现

1. 绪论

1.1课题存在的背景以及意义

21世纪是一个电子竞技的新时代，我们通过电竞愉悦身心，交流感情，在工作之余作为一种休憩的手段，体验者新的技术给我们带来的种种好处，在这其中，平台起到了不可或缺的作用，通过平台，我们总是能第一手的获取到最新的游戏资讯，通过平台，官方总是能及时的对出现的各种状况问题第一时间反馈给玩家，通过平台，玩家们能很好地交流彼此的新的，这给游戏带来了巨大的流量，同时也增强了游戏的趣味性，我们使用着如此便利的内容，然而我们却很少去思考平台的背后是什么内容在支撑着，一个好的平台应该是怎样的，一个好的平台应该有怎样的技术支持呢？从电子竞技诞生至今，从前端以及后端诞生至今，有哪些好的内容值得我们去关注，我们应该怎么利用这些内容区搭建一个好的在线交流平台，这将会是本文着重讨论的内容，本文将从前端内容以及相关的各种技术起手，来实现在线交流平台。

1.2 市场存在的平台现状

我们以腾讯的wegame平台为例，wegame平台不仅仅是一个玩家交流平台，更准确的称呼，wegame是一个综合性的平台，如果做一个集合的话，wegame>玩家游戏交流平台，但是虽然如此，wegame仍然在玩家交流这一方面做的可圈可点。首先就是它的咨询，轮播各种实施热点，再配合一些图片，能够较为清晰地表达出想表达的内容，比起其它传统的只有文字的表现形式，自然要更加吸引人，下面就是一些文字性的新闻内容以及一些近期的赛事，热门视频，搞笑视频，解说视频等等。翻开下一页，我们可以看到有商店，在商店里面，我们可以对我们玩的某款游戏做一个评价，或者去体验新的游戏，不足之处在于宣传内容和实际内容并不是完全符合，这就让人感觉大相径庭。总体来说目前的已存在的平台做的已经比较完善，我们可以借鉴做的好的地方，对于这些虚假宣传的内容我们要尽量抵制，尽量给玩家还原真实的环境。

1. 网站需求分析

首先我们从一张图中看出用户的需求

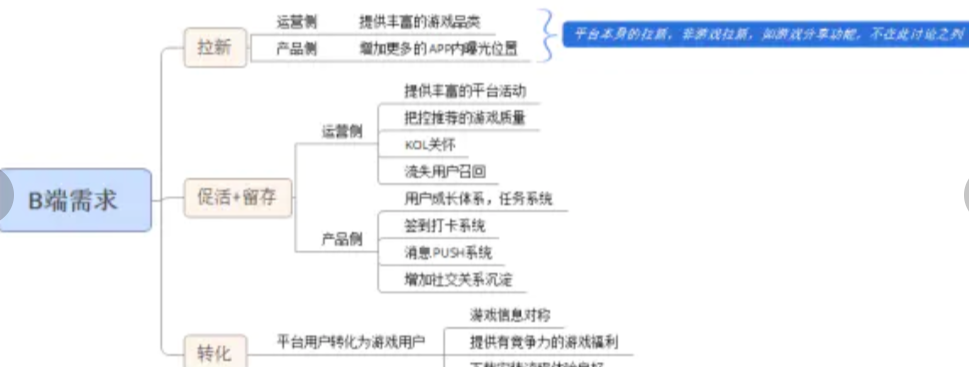


图2-1用户需求

根据马斯洛需求层次理论， 玩家的需求大致可以分为一下几类，基础需求，进阶需求，高级需求。其中基础需求是所有类型的玩家都必须满足的需求。因为如果没有基础需求其他的需求便都不存在，基础需求是进阶需求和高级需求的基础，但是进阶需求和高级需求比较重要，为了保证了游戏平台功能够持续的并且累积的活跃，形成一个完整生态闭环进阶需求和高级需求是关键。但是进阶需求和高级需求得视用户群体而定，如果平台游戏用户依旧为普通玩家，产品迭代重心应该还是在完善基础需求上面。

我们从玩家需要打发玩游戏时间这一需求上来分析



图2-2玩家需求

对应路径是这样的：

随便看看的类型：游戏平台不仅仅可以增加各类内容排行榜，并且可以增加各种个性化推荐（我们举个例子来说，比如基于地理位置推荐，更比如周围人在玩的游戏），游戏详情页，游戏分类等等这都是写好的思路。

查看最近新鲜的平台福利活动：福利活动是非常重要的一个环节，玩家们很有可能根据福利活动决定多玩或少玩游戏，对游戏的粘性有很大关系，首页增加福利模块，消息系统push重大福利活动

查看发布的帖子：比如说新增论坛，同时也需要支持查看历史回帖等消息

和其他的线上团体成员之间的交流：增加例如社群功能，支持即时通讯和短消息通讯等等

第3章 后端开发

3.1 技术框架设计

以下是整体的架构设计图

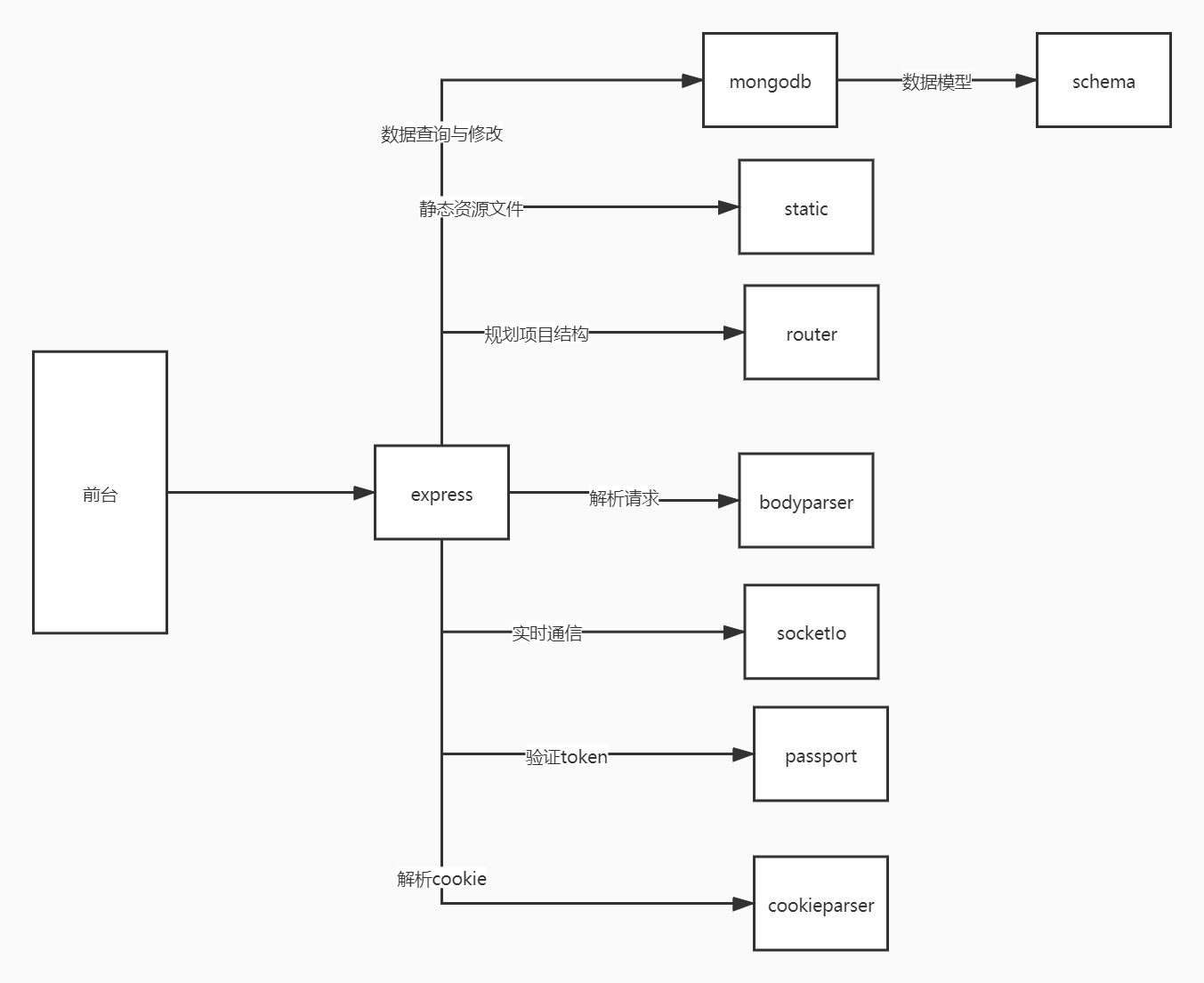


图3-1后端框架设计

前台的内容在此图中不作介绍，后端开发，首先是架构设计，架构上来说，首先要确定是使用express还是koa，鉴于对nodejs掌握更好，对koa接触较少，虽然在小程序开发过程中接触过koa，但是从整体上来看，国内社区更偏向于express，而且express由于其历史悠久，因此各种中间件非常完善，这大大降低了我们的开发难度，基于以上几点考虑因此选择nodejs，系统架构上后端一定要覆盖住所有的内容。

3.1.1 数据库设计

鉴于前端至少会实现登录注册，因此会涉及数据查询，配合mongodb，mongodb由c++编写，分布式文件存储数据库，鉴于其拥有超强的扩展性，因此使用起来非常方便。Mongodb通过schema来存储对象，相对于关系型数据库，这种存储的好处是更接近原生的数据类型，容易操作，效率不逊色与关系型数据库。在实际使用的过程中，我会将用户信息存储为一个schema，例如

const mongoose  = require('mongoose');

const Schema = mongoose.Schema;

//Create Schema

const UserSchema = new Schema({

    name: {

        type: String,

    },

    email: {

        type: String,

    },

    password: {

        type: String,

    },

    avatar: {

        type: String,

    },

    identity: {

        type: String,

    },

    date: {

        type: Date,

        default: Date.now

    }

})

module.exports = User = mongoose.model("users", UserSchema)

可以看到在这里我对用户设计了头像，用户可以通过上传图片作为自己的头像，日期作为用户的一个回忆事件，记录了用户第一次来到这里，用户看到这里会联想到自己在这里的点点滴滴，利用用户的回忆增强对平台的感情。

3.1.2 静态资源文件存储

Static是我们存储静态资源的位置，用户会想要存储自己收藏的一些图片资源，一些好的工具文件等等，我们所有的图片资源文件等等都会存储到这里

3.1.3 后端路由规划

router仅仅是对后端的目录的一个优化，主要为了方便开发人员，对路由进行分类，方便开发人员快速找到相应的内容，快速的对需要更改的项目进行修改。

3.1.4 前台内容解析

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。

3.1.5 实时通信

在玩家交流平台中，加入实时通信功能，会给用户带来更好的体验，媒体很重要的一点就是实时性非常重要，这会极大地增加客户的粘性，对于游戏玩家交流平台，这是必不可少的一部分内容，在评论这里，我们会加入实时讨论这一功能，这需要socketio的支持，

3.1.6 用户校验（1）

passport相对应的就是Jwt-oken,这是由于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。

3.1.7用户校验（2）

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。

3.2技术选型的原因：

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.单线程，3.非堵塞I/O，4.性能出众。Nodejs的事件机制是这样的

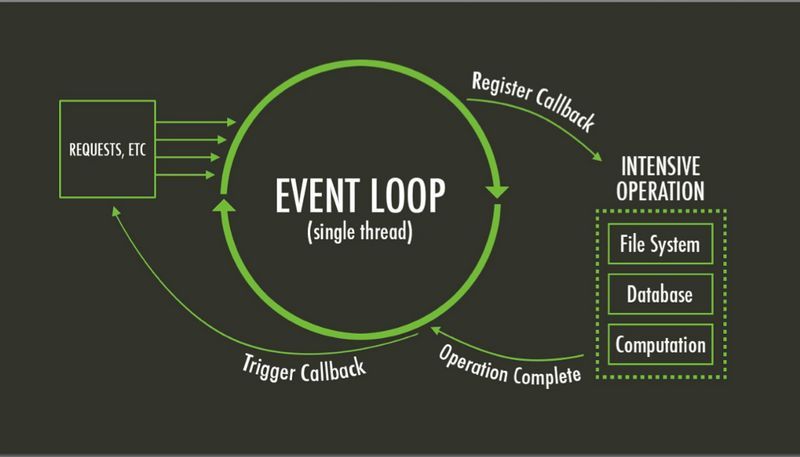


图3-2 nodejs事件机制

我们的场景类似于在线多人聊天，然而通过查阅文档我们知道，nodejs

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。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，再者是因为第三方的支持非常多。

第4章 前端UI设计

4.1设计ui

优美的界面非常重要，再开始写我们的项目之前，首先要设计好我们的ui，参考已有的游戏论坛平台，我设计了三分的结构

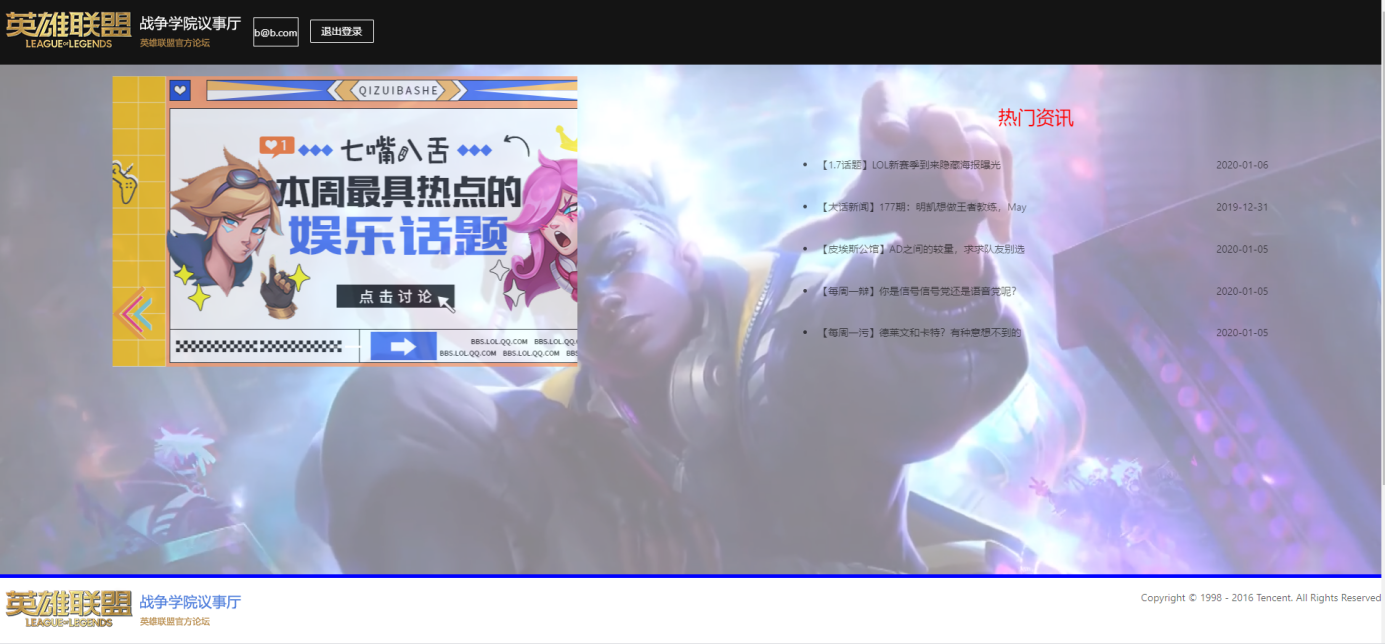


图4-1 前端页面

通过图片我们看到，三分的结构，好处主要是用户能非常方便的登入登出，通过顶层的navigation快速定位到自己想要查找或者操作的内容，同时，顶头的邮箱可以显示出用户的登入登出状态。

为了将最新的实时热点显示出来，我才用了轮播图，轮播图相比于单纯的文字展示，好处是通过图片，用户能够更清晰的知道此篇文章的内容是什么，同时提高了我们网站的浏览占比，充实了网站的内容，浏览的完整度变高，给用户提供了足够的优质信息。

但是由于轮播占的篇幅较大，并不能承载足够多的信息，于是我们在右侧增加tab来极大的丰富我们的内容，这是一种对称的网站布局

国人讲究对称美，这对于吸引我们的用户是大有益处的。

最后是所有官方网站都有的内容，网站的授权，以及网站的logo及网站的主要作用的标注。

第5章 前端程序开发

5.1技术选型

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。

对于数据框架我选择了mobx，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。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。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。

5.2前端开发

5.2.1技术框架设计

下面为我们的框架设计图

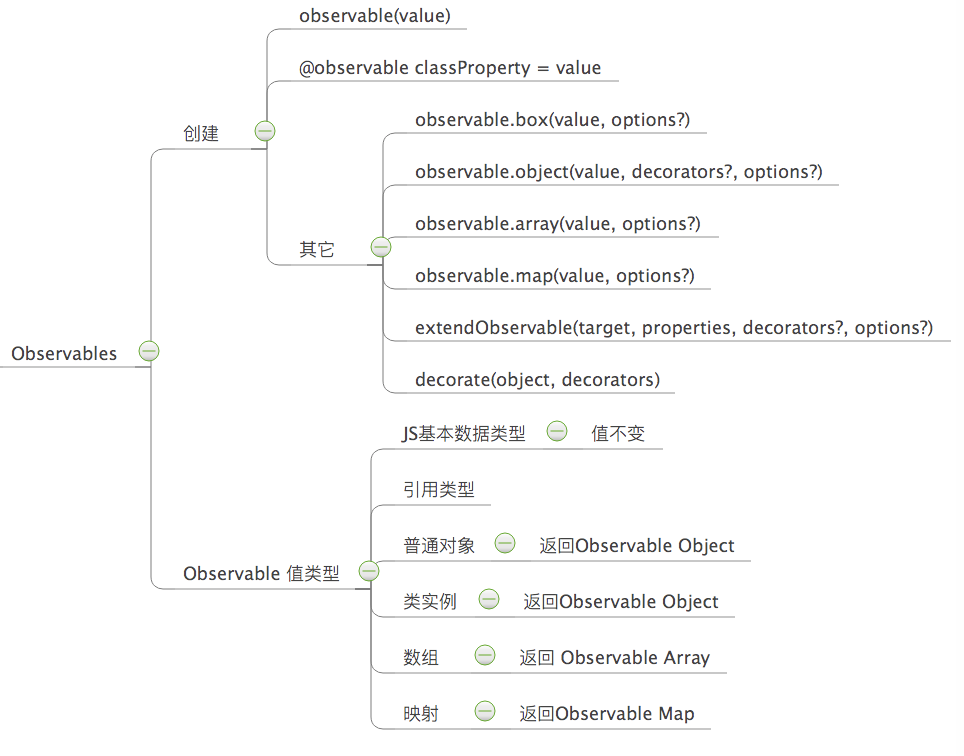


图5-1 前端框架设计

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。

5.2.2技术选型的原因

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。

5.2.3注册模块

1. 选取视频作为页面背景，此举是为了增加画面的优美程度，浏览过许多漂亮优美的网站，无非是动态，绚丽的界面能增加网站的体验，使用视频背景是最省时但是效果非常好的做法
2. 新建register.js，引用antd的form表单

import loadable from 'utils/loadable'//组件懒加载

import React, { useState, useEffect } from 'react';

import { withRouter } from 'react-router';

import { observer } from 'mobx-react-lite';

const BackgroundVideo = loadable(() => import('components/BackgroundVideo/BackgroundVideo'));

const NormalLoginForm = loadable(() => import('./components/NormalLoginForm/NormalLoginForm'));

export default observer(withRouter(function Login(props) {

    useEffect(() => {

    })

    return (

        <React.Fragment>

            <BackgroundVideo />

            <NormalLoginForm />

        </React.Fragment>

    )

}))

1. 发送请求到后端

export const register = (data) => {

    return axios.post('/api/users/register', data);

}

1. 后端接受请求并查看邮箱是否已经存在于数据库中，不存在则创建用户，存在则返回错误信息
2. 注册成功

成功

5.2.4个人中心

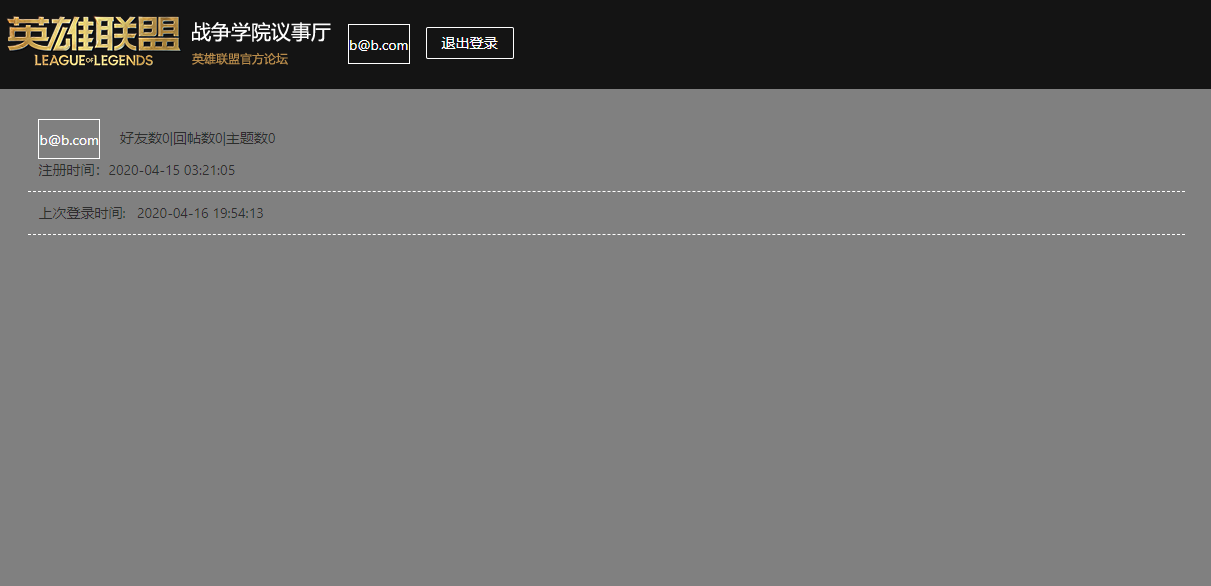


图5-2 个人中心

个人中心主要展示用户的信息，其内容有用户名，好友数量，回帖数量，主题数量，用户的注册时间以及用户的上次登录时间。在设计登录时间上遇到一些小的问题，一开始我在思考这个功能应该从何做起，首先用户登录必然会伴随着一个时间点，于是在用户登录后我自动的调用api存入当前的时间点，等到下次用户登录的时候，我再从数据库中自动查询，于是便完成了了这一功能，通过这个过程，让我明白了，遇到问题要一步步分析，不要因为刚开始不知所措而放弃，要慢慢探究整个过程，逐步分析出其逻辑。

5.2.5热点轮播

在设计轮播的时候，进入轮播页面后如何返回这个问题困扰了很久，直接点击浏览器的返回按钮这种方式似乎显得不是那么优雅，有可能我们输入的一些内容没有及时保存下来，通过对比其他交流平台的做法，我选择了点击我们的logo回到主页看到热点轮播的这种做法，热点轮播较为清晰地描述了当下的实时热点，见下图。



图5-3轮播图

点击热点轮播进入详情，详情页主要是玩家们对有关主题的讨论，玩家们分享自己的所见所闻所感，并且可以配上相应的图片，更好的表达玩家遇到此事或者此时的心情。见下图



图5-4 热点详情

5.2.6热门咨询

由于轮播图所表现出的内容有限，考虑到对称布局，因此在右侧增加了热门资讯板块，用来较为多的展示当下的实时热点，实时热点自然需要有实时性，因此对应的每一行都加上了咨询的时间，点击咨询我们能看到资讯里面的详细内容，主要是以文章为主，能够让玩家们方便的获取游戏知识，提升游戏技巧，使得玩家获得更好的游戏体验。

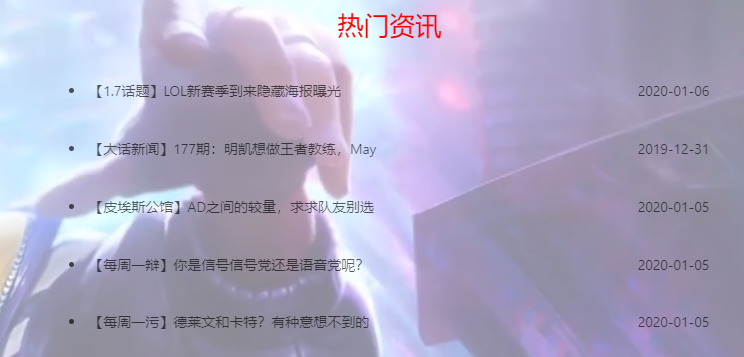


图5-5 热门咨询

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

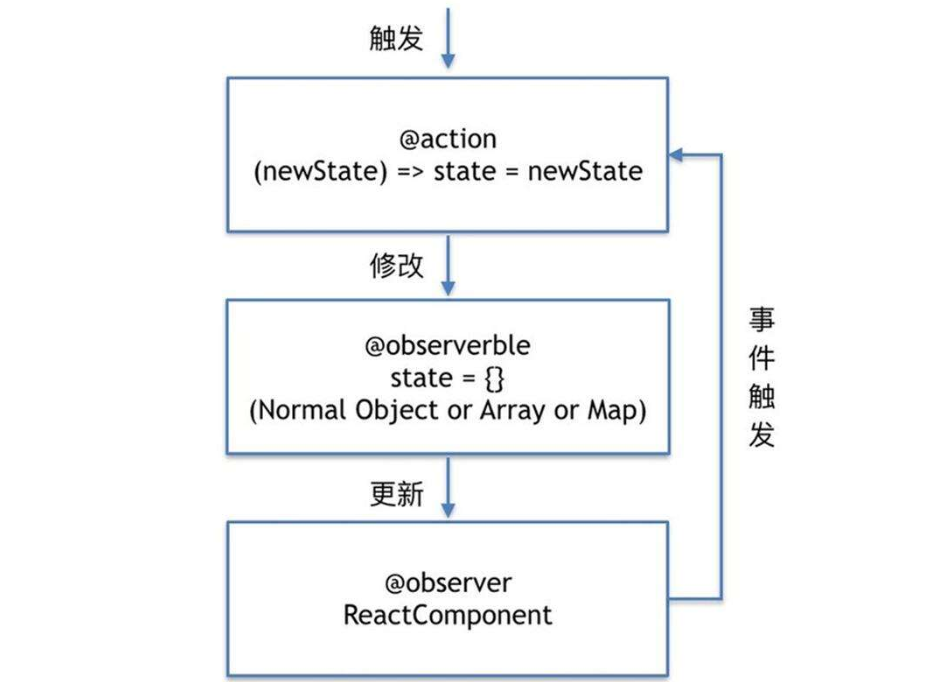


图6.1 mobx运行机制

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，在此过程中，前两次由于没有经验都出现了重大错误，章节不够，只是单纯的写了一些教程，不具有论文的性质，没有体现出自己的设计方案，而且格式也不太对，大部分时间在说各种工具与概念，这偏离了我的主题，应当写我在设计系统上的方案，架构，或者是遇到的问题，我是怎样解决这些问题的，通过对这些内容的仔细对比，最终经历多个版本的修改，终于完成，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。