

**毕业论文（设计）文献综述**

**[](http://bysj.zjgsu.edu.cn/)**

**题目：前端整合框架在交易系统中的应用**

**学 院：电子商务与工程管理学院**

**专 业：电子商务**

**班 级：商务16A**

**学 　 号：1612200132**

**学生姓名：王勇**

**指导教师：**

**二○ 一七 年 12月**

**《前端整合框架在交易系统中的应用》**

**文献综述**

**摘要：**在现代Web系统的开发的过程中，MVC模式中的View层交互功能越来越强大，业务逻辑也越来越复杂。传统的前端开发模式已经满足不了复杂的电子商务系统的业务需求的扩展与维护。为了使得前端应用的开发更加有效率，近些年出现了许多前端框架和自动化构建工具。本文在使用传统的技术方法实现电子商务系统的前端页面的提前下，使用前端整合框架React.js+ Redux + Reat-Router对该系统前端页面进行模块化重构，由此比较两种开发模式的主要区别。

**关键词:**Web前端,，React.js，Redux, 单页应用，JSX，单向数据流，电子商务系统前端，前端模块化开发

**“The Application of Front - end Integration Framework in E - commerce System”**

**Abstract:** In the development of modern Web system, interaction in view layer becomes more and more powerful and business logic is also more and more complex. The traditional front-end development model has been unable to meet the complex e-commerce system’s business needs expansion and maintenance. In order to make the development of front-end applications more efficient, in recent years there have been many front-end framework and automated building tools. In this paper, the traditional method is used to build the e-commerce system in advance, using the front-end integration framework React.js to modulate the front-end applications of the system, thus comparing the main differences between the two development models.

**Keywords:** Web front end; React.js; Redux; single-page-application; Single data stream; front-end reconstruction; front-end modular development

# 一、引言

随着互联网的发展，前端交互的场景变得越来越复杂，为了解决这些在前端开发中的痛点，在最近几年的前端领域中，优秀的框架层出不穷，比如Vue.js, AngularJS使用web前端框架搭建的单页应用（SPA）在不向服务器发起HTTP请求的情况下就可以实现路由的转发，并且按需加载所需要的文件，极大的提升了用户的体验。正是因为React技术栈实现了前端架构的高内聚，低耦合，国内很多大型互联网公司包括阿里巴巴和蚂蚁金服的部分的业务也使用React进行了前端的重构。在实现后台系统和传统JQuery构建前端应用的前提下，如何通过前端框架的整合，将传统方式构建的web电子商务前端页面进行重构，实现前端的状态管理并且兼容移动端的单页应用是本次研究需要解决的问题。

# 二、传统web前端开发

## （一）开发流程

在早期的web时代，人们使用浏览器大多都是用浏览内容，充其量填写一些表单传送到服务器，传统web应用的开发，将不同的种类的技术分门别类。放到不同的文件夹中，先编写HTML页面，确定页面的语义，再编写CSS代码实现页面布局，最后编写JavaScript实现 页面的交互。这样的分工并没有做到逻辑上的分治。只是单纯把不同类型的技术分开了而已。

## （二）存在的问题

在传统的web前端运用的开发中,开发人员常常会使用jQuery在DOM节点上绑定事件，一旦用户与界面的交互变得复杂之后，事件和元素的关系会变得纠缠不堪。导致代码难以维护，功能难以扩展，状态难以管理。而且在DOM节点绑定大量的事件很容易造成内存的泄露。

为了加强代码的复用性，在开发的过程中常常会封装大量的组件，但是组件的样式又部分在CSS文件夹中，导致了组件复用的复杂程度。而且在使用jQuery编写组件的过程中会多次操作DOM节点，严重影响性能。

View层直接和处理数据的业务逻辑交互，导致处理相同逻辑业务代码在重构代码的时候才能抽离出来，降低了代码的复用，严重影响开发的效率。在后期需求发生变化的时候，可能修改一个部分就会导致其他的bug.

但是随着前端页面的交互逻辑变得越来越复杂的时候，更多的数据处理和业务的逻辑交给了客户端去处理，可能一个网站需要维护的javaScript代码可能到达上万行，在保持用户体验，减少http请求的情况下，一个页面的所有的逻辑都会在一个javaScript文件中。极其不利于代码的扩展与复用。

## （三）MVC框架

为了克服这些问题，出现了前端的MVC框架，在MVC框架中我们把应用分成了三个部分：Model来管理数据，View层来负责渲染用户的界面，Controller来负责接收用户的响应，并根据用户的行为来改变Model，再把产生的数据结果交给View层，刷新用户界面。

但是在前端MVC框架实现中，并不一定有Controller, 客户端根据用户的行为修改客户端Model,客户端更新和该Model相关的View, 客户端更新和该Model送sync请求到服务器，服务器返回修改是否成功。如果不成功，返回上一个view的显示结果。所以前端的MVC框架并不是严格意义上遵循MVC的，常常为了开发的便捷，View层和Model层会直接进行通信。这就是导致了，View层和Model的逻辑会交错复杂。

# 三、React.js

FaceBook开源React.js引起了业界极大的关注。由于FaceBook前端业务需求越来越复杂，MVC的前端框架已经满足不了其复杂的业务场景的需要，FaceBook推出了React + Flux 来解决前端复杂的交互场景。

## 数据驱动渲染

在React的理念可以归结为一个简单的公式： [7] UI = render（data）

这里render可以说是一个纯函数，那么它的返回结果只会受参数data的影响，所以只要data不发生变化，那么UI就不会发生变化。所以在React中我们所能看到的UI是完全取决于data的状态的，只有data发生改变的时候，UI才会更新，所以react也实践了响应式编程的思想。那么开发者只要思考，需要在组件中显示什么样的数据就可以了，不需要思考具体要怎么样去显示。

那么我们编写交互的方式也从操纵DOM元素转变成了操纵数据，在react中分别有props和state保存着组件的数据也就是上文所提到的data参数，props是来自外界的数据，state是组件自身维持的数据。每当props和state发生变化时就会引起组件的重新渲染。

## （二）Virtual DOM

显而易见的是React如果发现组件的数据进行了变化，通过重复渲染组件来改变UI。但是通常情况下，一个组件中只有一部分的数据发生了变化，如果重新渲染整个组件无疑是浪费时间的。

React采用了Virtual DOM来解决这个问题，Virtual DOM 采用了React Diff算法，在组件状态发生了变化之后。React会在先在JavaScript树形结构中构建Virtual DOM, 每次渲染组件之前都会和之前的渲染结果对比，然后发现不同的处，在真正修改DOM时候，就只要修改有差别的部分就足够了。

## （三）与传统开发相比React的优势

传统的web开发通过直接操控DOM节点，这种方式很容易上手，开发者只要改变DOM节点的属性就能够改变用户界面的UI 。当项目变得庞大的时候，如图1所示，用JQuery写出的代码往往和DOM节点互相纠缠，当节点的事件和引用没有清理的情况下是很容易引起内存泄漏的。

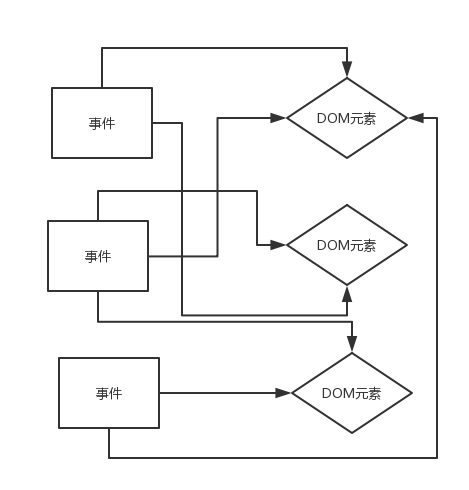


图1

React通过函数式编程的思维，通过了数据来驱动UI的改变，强制要求每一个组件，必须通过数据来驱动渲染，所以不管应用多么庞大复杂，都能让程序处于一个可控的范围。

React让开发者专注于界面应该是显示成什么样子的，而不是怎么样去显示这个界面。大大提高了开发组织代码的效率。

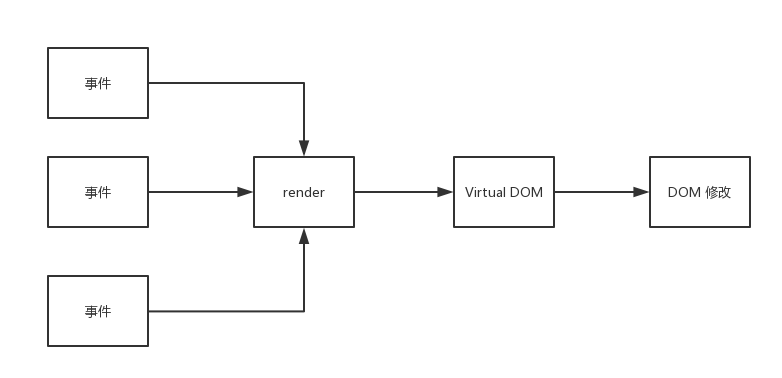


图2

# 四、Vue.js

## （一）渐进式框架

在框架使用的过程中，不免会被框架框架自身所限制。每个框架都有自身的主张，对比React和AngularJS我们就会发现，React主张的是函数式编程。而AngularJS更具有排他性，一旦使用AangularJS，就必须使用它的模块机制、依赖注入和特定的组件。而Vue采用渐进式的设计原则，框架的使用者可以按照需求来进行配置，从简单到复杂的场景都可以覆盖到。

## 双向数据绑定

在Vue中通过使用 ES5 提供的 Object.defineProperty() 方法，监控对数据的操作，从而可以自动触发数据同步，实现了数据双向绑定。并且，由于是在不同的数据上触发同步，可以精确的将变更发送给绑定的视图，而不是对所有的数据都执行一次检测，避免了繁琐的DOM操作。[10]每个Vue实例都会有相应的watcher实例对象。如果data数据源发生了变化，就会通知到watcher从而比较Virtual DOM 异步更新组件中需要更新的部分。

# 五、AngularJS

## （一）MVC

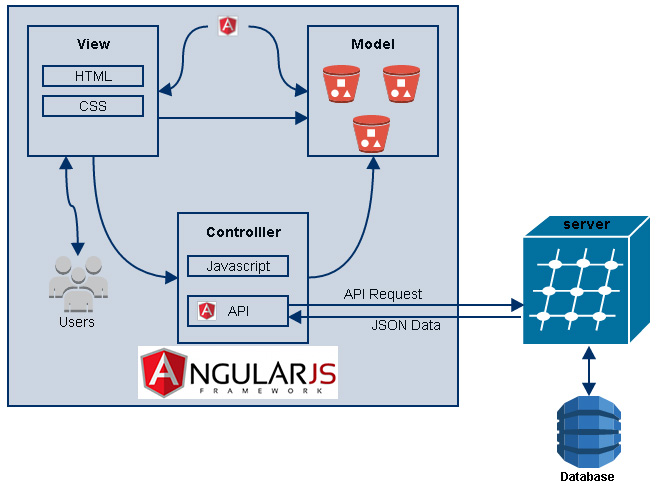


图3

如图3所示与React和Vue相比，Angular会显得十分的笨重。也只有Angular集成了MVC的每一个环节。

## （二）双向数据绑定

AngularJS 采用“脏值检测”的方式，数据发生变更后，对于所有的数据和视图的绑定关系进行一次检测，识别是否有数据发生了改变，有变化进行处理，可能进一步引发其他数据的改变，所以这个过程可能会循环几次，一直到不再有数据变化发生后，将变更的数据发送到视图，更新页面展现。如果是手动对 ViewModel 的数据进行变更，为确保变更同步到视图，需要手动触发一次“脏值检测”。

# 六、前端模块化和工程化

有了这几大框架，应对大型前端应用的开发似乎已经很完美了，但是考虑到团队的协作，似乎还是存在着许多的问题。例如：如何统一使用的框架的版本，如何制定开发的代码规范，如何处理各个模块之间的依赖关系。前端工程化就是为了解决如上所说的这些问题。

工程化的要素，会有以下几个方面

* 依赖管理与统一的组件发布与仓库

npm依赖于Node.js平台，是现代前端应用的开发的依赖管理工具。通过package.json配置文件的定义就可以统一开发环境对不同框架依赖的版本。开发者也可以通过npm平台管理和维护自己构建的工具库和插件，使得代码能够重复利用。

* 统一的开发规范

对于代码的规范需要人为制定规则和统一。可以通过ESLint制定代码规范，ESlint中定义了200多条JavaScript的编码规范。我们可以通过配置文件，决定代码缩进的格式，避免变量隐式转换，禁止扩展原生对象的原型，禁止使用with语句。

* 模块化/组件化开发

1. 模块化

随着前端代码的复杂程度不断地提高，JavaScript模块化的概念被提了出来，在没有模块化的前，编写JavaScript代码时，往往要在一个html页面中引入多个<script>标签。这样的方式组织代码会导致依赖关系不明显，也不利于代码后期的维护。

随之提出的AMD和CMD规范就是为了解决在浏览器端模块化开发的问题，依托了加载器实现的javascript模块化开发和加载，极大程度地提升了前端开发的体验。

1. Webpack

但是模块化开发的体验对开发人员来说是好，但是在加载上就有一些问题， 在前端优化上，我们一般会通过减少请求来优化页面的性能， 模块化加载显然是背道而驰的。这时候解决的方案就是把模块进行打包，发布前把所有的依赖打包在一起，这样就可以把页面资源请求的降到了最低。这时候打包工具就出现了，当前Webpack成为了前端应用最佳的依赖打包的解决方案，成为了当前最流行的依赖打包工具。在Webpack中，通过loader处理过的一切文件都是模块。Webpack还能够处理JavaScript和CSS扩展语言，将这些文件转换成在浏览器端能够使用的文件。Webpack还提供了插件的形式，支持拓展的功能。只要通过npm进行下载，在配置文件进行插件的配置。

# 七、参考文献

[1]CORY GACKENHEIMER著；张铮铮译.REACT导学[M].北京：人民邮电出版社.2016.

[2]（英）Artemij Fedosejev著.React 精髓[M].北京：电子工业出版社.2016.

[3]张轩著.React全栈 Redux+Flux+webpack+Babel整合开发[M].北京：电子工业出版社.2016.

[4]刘一奇.React与Redux开发实例精解[M].北京：电子工业出版社.2016.

[5]程墨著.深入浅出React和Redux[M].北京：机械工业出版社.2017.

[6]封宇[1,2],陈宁江[1].基于MVVM架构的移动Web前端展案[J].计算机与现代化,2014,0(11):1-4.

[7] 孙连山,李云倩.MVVM框架在Web前端的应用研究[J].电脑知识与技术：学术交流,2016,0(2X):45-46.

[8] [1]冯晨超. 基于AngularJS的物品和用户查询模块的设计与实现[D].南京大学,2014.