前端框架方案

目 录

[第一章 Web前端 5](#_Toc445108367)

[第一节 概述 5](#_Toc445108368)

[第二节 前沿思想 5](#_Toc445108369)

[一、 SPA 5](#_Toc445108370)

[二、 组件化开发 5](#_Toc445108371)

[三、 RESTFul 5](#_Toc445108372)

[四、 工程化 5](#_Toc445108373)

[第三节 主流框架 5](#_Toc445108374)

[一、 AngularJs 5](#_Toc445108375)

[二、 ReactJs 5](#_Toc445108376)

[三、 Vue 5](#_Toc445108377)

[第二章 AngularJs 5](#_Toc445108378)

[第一节 概述 5](#_Toc445108379)

[第二节 核心思想 5](#_Toc445108380)

[一、 MVVC 5](#_Toc445108381)

[二、 依赖注入 6](#_Toc445108382)

[三、 双向绑定 6](#_Toc445108383)

[第三节 主要功能 6](#_Toc445108384)

[一、 作用域 6](#_Toc445108385)

[二、 路由 6](#_Toc445108386)

[三、 模板引擎 6](#_Toc445108387)

[四、 支持单元测试和端对端测试 6](#_Toc445108388)

[第四节 问题和缺陷 6](#_Toc445108389)

[一、 性能 6](#_Toc445108390)

[二、 学习曲线 6](#_Toc445108391)

[第三章 代码结构和规范 6](#_Toc445108392)

[第一节 项目结构 6](#_Toc445108393)

[第二节 代码类别 6](#_Toc445108394)

# Web前端

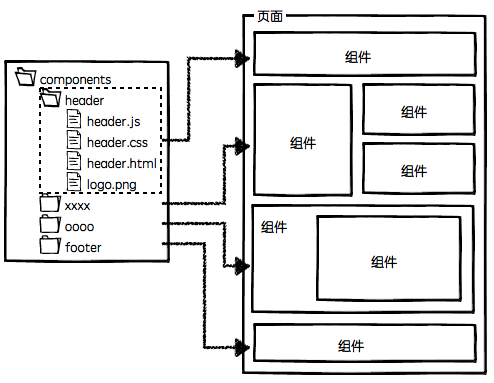
## 概述

随着时代的发展，前端从最初单纯的页面展示，转变为了具有多样化的展现方式、复杂的交互方式和更重视用户体验的app。这也导致前端代码量大增、团队扩大，随之而来的团队协作、代码管理、代码质量等问题也越来越影响开发效率。那么使用框架也是顺理成章的事情，通过框架来引入类似模块化、mvc等编程思想，完善代码规范，进行单元测试，使前端更容易管理和优化。

## 前沿思想

### 组件化开发

组件化包括对数据逻辑层业务梳理，形成不同层级的能力封装。它不一定是一个自定义语义标签：它可以是一个包含逻辑(js)、样式(css)、模版(html)的功能完备的结构单元，也就是我们常“口口相传”的模块；它也可以是一个单纯的js，比如http组件这种纯逻辑单元。



简单理解组件化开发即是把整个页面切为一个一个的组件，每个组件相互独立，并且有对应的目录存放组件的资源。

由于组件间相互独立，可以根据需求随意替换调整组件，不会对其他部分造成影响。而且这种“分而治之”的思想也更容易分配工作，提高团队协作的能力。

### SPA

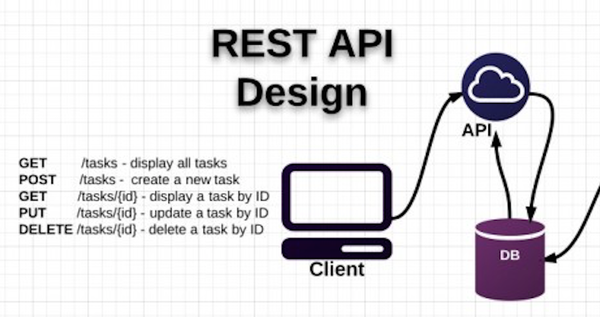
SPA（Single Page Application），即是单页面应用。只有一个页面，采用js控制，不重载页面，只刷新部分页面元素。

SPA在切换页面只需要加载部分html模板和json字符串，大大增加了响应速度，还减少了服务器的压力和带宽消耗。另外SPA更好使前后端高度分离，同一套后端能应对多个平台。

但SPA不利于SEO，而且前进后退等功能需要自己实现。

### REST

Representational State Transfer，这是一种架构风格，目标是构建可扩展的web service。REST规范可以提高架构的性能和可维护性。RESTful（采用REST架构规范的）系统通常是通过HTTP协议，并且使用HTTP的GET,POST,PUT,DELETE等动词来收发数据。



### 工程化

前端工程化就是以系统化、 规范化、 数量化等工程原则和方法去进行前端开发和维护。

首先，采用能最大程度帮助代码管理和团队协作的项目结构构建项目。

其次，统一编码风格和规范。

第三，在部署前对代码进行质量检测和单元测试。

最后，进行代码优化处理，采用利于运行的目录结构，部署项目。

### FLUX

Flux是Facebook用来构建用户端的web应用的应用程序体系架构。它通过利用数据的单向流动为可复用的视图组件提供了补充。Flux应用主要包括四部分：dispatcher、Action、store和views。



FLUX的核心特性是单向数据流，解决了传统 MVC框架下View和Model双向绑定导致的关系混乱。

## 前端框架

框架是一种特殊的、已经实现了的WEB应用，你只需要对它填充具体的业务逻辑。这里框架是起主导作用的，由它来根据具体的应用逻辑来调用你的代码。

框架有三个优点：首先，框架可以让程序员更专注于业务；其次，框架有较为完善的规范，能拉平整个团队的技术水平；最后，框架会在底层处理部分浏览器的差异，能提高web app的浏览器兼容性。

## 主流前端框架

### ReactJs

React 是一个Facebook和Instagram用来创建用户界面的JavaScript库，用于解决构建随着时间数据不断变化的大规模应用程序。React的核心内容：

* virtual dom：react中的组件跟页面真实dom之间会有一层虚拟dom。
* 单向数据流：flux架构下的数据流呈现出一种单向、闭环的流动路线，使得一切行为变的可预测，也能更好的定位错误发生点。
* 状态机：每个组件都是状态机，组件关注的只应该是状态，不同的状态呈现不同的表现形式。
* 组件：一切都是组件。
* Immutable object：使用只读数据来建立数据模型，每次更新都是new object。

### Vue.js

Vue.js是国内一个用于创建 web 交互界面的库，是一个精简的MVVM。vue专注于 MVVM 模型的 ViewModel 层，通过双向数据绑定把 View 层和 Model 层连接了起来。

相较其他mv\*框架，vue轻量、易用、更优异的性能及面向组件化的设计。而且阿里手淘团队发布了名为“无线电商动态化解决方案”的 Weex，该方案就是基于vue.js。

# AngularJs

## 概述

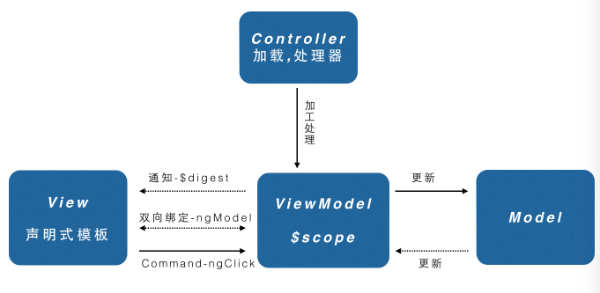
AngularJs是一款开源JavaScript库，由Google维护，用来协助单一页面应用程序运行的。框架采用并扩展了传统HTML，通过双向的数据绑定来适应动态内容，双向的数据绑定允许模型和视图之间的自动同步。因此，AngularJS使得对DOM的操作不再重要并提升了可测试性。

AngularJs更适合CRUD应用，例如阿里云管理控制台。而对于dom操作频繁且复杂的应用，像游戏或图形界面编辑器，就不太适合使用AngularJs。

## 核心思想

### MVVM

MVVM模式是Model-View-ViewMode（模型-视图-视图模型）模式的简称，是MVC模式的衍生物。



View负责于界面的显示和渲染，在angular中则是包含一堆声明式Directive的html模板。

Model是与应用程序的业务逻辑相关的数据的封装载体，Model并不关心会被如何显示或操作，所以Model也不会包含任何界面显示相关的逻辑。在Angular中的Service则是封装和处理这些与Model相关的业务逻辑的场所，这类的业务服务是可以被多个Controller或者其他Service复用的领域服务。

ViewModel是View和Model的粘合体，负责View和Model的交互和协作，它负责给View提供显示的数据，以及提供了View中Command事件操作Model的途径；在angular中$scope对象充当了这个ViewModel的角色；

Controller：这并不是MVVM模式的核心元素，但它负责ViewModel对象的初始化，它将组合一个或者多个service来获取业务领域Model放在ViewModel对象上，使得应用界面在启动加载的时候达到一种可用的状态。

### 依赖注入

### 双向绑定

## 主要功能

### 作用域

### 语义化标签

### 路由

### 模板引擎

### 支持单元测试和端对端测试

## 与jquery的区别

## 问题和缺陷

### 性能

### 学习曲线

# 代码结构和规范

## 项目结构

## 代码类别