AngularJS 开发者最常犯的 10 个错误

12
顶

英文原文：The Top 10 Mistakes AngularJS Developers Make
标签： AngularJS
oschina 推荐于 10个月前 (共 16 段, 翻译完成于 10-07) (24评)

参与翻译(5 LeoG0816, 砼砼, 柳絮飞_lilf, BreakingBad, James_陋

人)：

介绍

AngularJS是如今最受欢迎的JS框架之一，简化开发过程是它的目标之一，这使得它非常适合于元型较小的apps的开发，但也扩展到具有全部特征的客户端应用的开发。易于开发、较多的特征及较好的效果导致了较多的应用，伴随而来的是一些陷阱。本文列举了AngularJS的一些共同的易于也问题的地方，尤其是在开发一个app的时候。

翻译的不错

柳絮飞_lilf
顶 哦!10个月前

1人顶

1. MVC目录结构

AngularJS是一个缺乏较好的term的MVC框架，其models不像backbone.js中那样做为一个框架来定义，但其结构模式仍匹配的较好。当在一个MVC框架中作业时，基于文件类型将文件组合在一起是其共同的要求：

```
templates/
    _login.html
    _feed.html
app/
    app.js
    controllers/
        LoginController.js
        FeedController.js
    directives/
        FeedEntryDirective.js
    services/
        LoginService.js
        FeedService.js
    filters/
        CapatalizeFilter.js
```

翻译的不错

BreakingBad
顶 哦!10个月前

1人顶

这样的布局, 尤其是对那些有 Rails 背景的人来说, 看起来挺合理. 可是当 app 变得越来越庞大的时候, 这样的布局结构会导致每次都会打开一堆文件夹. 无论你是用 Sublime, Visual Studio, 还是 Vim with Nerd Tree, 每次都要花上很多时间滑动滚动条浏览这个目录树来查找文件.

如果我们根据每个文件隶属的功能模块来对文件分组, 而不是根据它隶属的层:

```
app/
    app.js
    Feed/
        _feed.html
        FeedController.js
        FeedEntryDirective.js
        FeedService.js
    Login/
        _login.html
        LoginController.js
        LoginService.js
    Shared/
        CapatalizeFilter.js
```

那么查找某个功能模块的文件就要容易得多, 自然可以提高开发的速度. 也许把 html 文件跟 js 文件放在混合放在一起做法不是每个人都能认同. 但是起码它省下宝贵的时间.

翻译的不错

James_陋
顶 哦!10个月前

1人顶

2、模块分组

一开始就将主模块中所有子模块展示出来是通常的做法。但是开始做一个小应用还好，但是做大了就不好管理了。

```
var app = angular.module('app',[]);app.service('MyService', function(){
    //service code});app.controller('MyCtrl', function($scope, MyService){
    //controller code});
```

一个比较好的办法是将相似类型的子模块分组：

```
var services = angular.module('services',[]);services.service('MyService', function(){
    //service code});var controllers = angular.module('controllers',['services']);control
    //controller code});var app = angular.module('app',['controllers', 'services']);
```

这个方法与上面那个方法效果差不多，但是也不很大。运用要分组的思想将使工作更容易。

```
var sharedServicesModule = angular.module('sharedServices',[]);
sharedServices.service('NetworkService', function($http){});
var loginModule = angular.module('login',['sharedServices']);
loginModule.service('loginService', function(NetworkService){});
loginModule.controller('loginCtrl', function($scope, loginService){});
var app = angular.module('app', ['sharedServices', 'login']);
```

当创建一个大的应用时，所有模块可能不会放在一页里，但是将模块根据类型进行分组将使模块的重用能力更强。

3 依赖注入

翻译的不错

LeoG0816
顶 哦!10个月前

1人顶

依赖注入是AngularJS最棒的模式之一。它使测试变得更加方便，也让它所依赖的对象变的更加清楚明白。AngularJS 对于注入是非常灵活的。一个最简单的方式只需要为模块将依赖的名字传入函数中：

```
var app = angular.module('app',[]);app.controller('MainCtrl', function($scope, $timeout){
    $timeout(function(){
        console.log($scope);
    }, 1000);});
```

这里，很清楚的是MainCtrl依赖于$scope和$timeout。

直到你准备投入生产并压缩你的代码。使用UglifyJS，上面的例子会变成：

```
var app=angular.module("app",[]);
app.controller("MainCtrl",function(e,t){t(function(){console.log(e)},1e3)})
```

现在AngularJS怎么知道MainCtrl依赖什么？AngularJS提供了一个非常简单的解决方案：把依赖作为一个字符串数组传递，而数组的最后一个元素是一个把所有依赖作为参数的函数。

```
app.controller('MainCtrl', ['$scope', '$timeout', function($scope, $timeout){
    $timeout(function(){
        console.log($scope);
    }, 1000);}]);
```

接下来在压缩的代码中AngularJS也可以知道如何找到依赖：

```
app.controller("MainCtrl",["$scope","$timeout",function(e,t){t(function(){console.log(e)
```

3.1 全局依赖

翻译的不错

LeoG0816
顶 哦!10个月前

1人顶

通常在写AngularJS应用时会有一个对象作为依赖绑定到全局作用域中。这意味着它在任何AngularJS的代码中都可用，但这打破了依赖注入模型同时带来一些问题，特别是在测试中。

AngularJS把这些全局变量封装到模块中，这样它们可以像标准AngularJS模块一样被注入。

Underscore.js是很棒的库，它把Javascript代码简化成了函数模式，并且它可以被转化成一个模块：

```
var underscore = angular.module('underscore', []);underscore.factory('_', function() {
  return window._; //Underscore must already be loaded on the page});var app = angular.mo
    init = function() {
          _.keys($scope);
      }

      init();}]);
```

它允许应用继续用AngularJS依赖注入的风格，也让underscore在测试的时候被交换出来。

这或许看上去不重要，像是一个无关紧要的工作，但如果你的代码正在使用use strict（应该使用），那么这就变得有必要了。

4 控制器膨胀

翻译的不错

LeoG0816
顶 哦!10个月前

1人顶

控制器是AngularJS应用中的肉和番茄。它很简单，特别是开始的时候，在控制器中放入过多的逻辑。控制器不应该做任何DOM操作或者有DOM选择器，这应该由使用ngModel的指令（directives）做的事。同样地，业务逻辑应该在服务（services）中，而不是 控制器。

数据也应该被存在服务（services）中，除非它已经和$scope关联。服务（services）是留存于整个应用生命周期的个体，同时控制器在应用各阶段间都是暂态的。如果数据被存在控制器中，那么当它被重新实例化的时候，就需要从其他地方抓取。即使数据被存储在localStorage中，获取数据也要比从Javascript变量中获取要慢几个数量级。

AngularJS在遵从简单责任原则（SRP）时工作地最好。如果控制器是视图和模型的协调者，那么它拥有的逻辑应该被最小化。这将使得测试变的更加简单。

5 Service 和 Factory的区别

几乎每一个刚接触AngularJS的开发者，都会对这两个东西产生困惑。 虽然它们（几乎）实现了同样的效果，但真的不是语法糖。

这里是它们在 AngularJS 源码中的定义:

```
function factory(name, factoryFn) { return provider(name, { $get: factoryFn }); }

function service(name, constructor) {
    return factory(name, ['$injector', function($injector) {
      return $injector.instantiate(constructor);
    }]);
  }
```

从源码上看显然 service 函数只是调用 factory 函数，然后 factory 函数再调用 provider 函数。事实上，value、constant和decorator 也是 AngularJS 提供的对 provider 的封装，但对它们使用场景不会有这种困惑，并且文档描述也非常清晰。

那么Service 仅仅是单纯的调用了一次 factory 函数吗？ 重点在 $injector.instantiate 中; 在这个函数里service会接收一个由$injector 使用new关键字去实例化的一个构造器对象。（原文：with in this function $injector creates a new instance of the service's constructor function.）

翻译的不错

砂砂
顶 哦!10个月前

2人顶

下面是完成同样功能的一个service和一个factory。

```
var app = angular.module('app',[]);

app.service('helloWorldService', function(){
    this.hello = function() {
        return "Hello World";
    };});

app.factory('helloWorldFactory', function(){
    return {
        hello: function() {
            return "Hello World";
        }
    }});
```

翻译的不错

砂砂
顶 哦!10个月前

2人顶

当 helloWorldService 或者 helloWorldFactory中的任何一个注入到controller里面, 他们都有一个返回字符串"Hello World"的名称为 hello方法。 这个service 的构造函数只在声明时被实例化一次，并且在这个 factory 对象每次被注入时各种互相引用， 但这个 factory还是只是被实例化了一次。 **所有的 providers 都是单例的。**

既然都完成同样的功能，为什么会有这两种格式存在？factory比service略微更灵活一些，因为它们可以使用new关键字返回函数（原文：Factories offer slightly more flexibility than services because they can return functions which can then be new'd）。 在其他地方，从面向对象编程的工厂模式来说。 一个factory可以是一个用于创建其他对象的对象。

```
app.factory('helloFactory', function() {
    return function(name) {
        this.name = name;

        this.hello = function() {
            return "Hello " + this.name;
        };
    };
});
```

这里有一个使用了前面提到的那个service和两个factory的controller 的例子。需要注意的是 helloFactory 返回的是一个函数，变量name的值是在对象使用new关键字的时候设置。

```
app.controller('helloCtrl', function($scope, helloWorldService, helloWorldFactory, hel
    init = function() {
      helloWorldService.hello(); //'Hello World'

      helloWorldFactory.hello(); //'Hello World'

      new helloFactory('Readers').hello() //'Hello Readers'

    }
    init();
});
```

翻译的不错

砂砂

在刚入门时候最好只使用services.

Factory更加适用于当你在设计一个需要私有方法的类的时候使用：

```
app.factory('privateFactory', function(){
    var privateFunc = function(name) {
```

```
        return name.split("").reverse().join(""); //reverses the name
    };

    return {
        hello: function(name){
          return "Hello " + privateFunc(name);
        }
    };});
```

在这个例子中privateFactory含有一个不能被外部访问的私有privateFunc函数。这种使用方式services也可以实现，但是使用Factory代码结构显得更加清晰。

6 不会使用 Batarang

Batarang 是用于开发和调试 AngularJS 应用的一个优秀的chrome浏览器插件。

Batarang 提供了模型浏览，可以查看Angular内部哪些模型已经绑定到作用域（scopes）。可以用于需要在运行时查看指令中的隔离作用域（isolate scopes）绑定的值。

Batarang 还提供了依赖关系图。 对于引入一个未测试的代码库， 这个工具可以快速确定哪些services应该得到更多的关注。

最后， Batarang提供了性能分析。 AngularJS 虽然是高性能开箱即用, 但是随着应用自定义指令和复杂的业务逻辑的增长，有时候会感到页面不够流畅。使用 Batarang 的性能分析工具可以很方便的查看哪些functions 在digest 周期中占用了更多的时间。这个工具还可以显示出整个监控树（full watch tree），当页面有太多的监控器（watch）时，这个功能就显得有用了。

顶 哦! 10个月前

2人顶

7 太多的watchers

翻译的不错

柳絮飞_lilf

顶 哦! 10个月前

2人顶

正如上文中提到的，在外部AngularJS是很不错的。因为在一个循环消化中需要进行dirty检查，一旦watcher的数目超过2,000，循环会出现很明显的问题。（2,000仅是一个参考数，在1.3版本中AngularJS对循环消化有更为严谨的控制，关于这个Aaron Graye有较为详细的叙述）

这个IIFE（快速响应函数）可输出当前本页中的watcher的数目，只需将其复制到console即可查看详情。IIFE的来源跟Jared关于StackOverflow的回答是类似的。

```
(function () {
    var root = $(document.getElementsByTagName('body'));
    var watchers = [];

    var f = function (element) {
        if (element.data().hasOwnProperty('$scope')) {
            angular.forEach(element.data().$scope.$$watchers, function (watcher) {
                watchers.push(watcher);
            });
        }

        angular.forEach(element.children(), function (childElement) {
            f($(childElement));
        });
    };

    f(root);

    console.log(watchers.length);})();
```

使用这个，可以从Batarang的效率方面来决定watcher及watch tree的数目，可以看到在哪些地方顾在或哪些地方没有改变的数据有一个watch。

当有数据没有变化时，但在Angular中又想让它成为模板，可以考虑使用bindonce.Bindonce在Angular中仅是一个可能使用模板的指令，但没有增加watch的数目。

8 审视$scope

翻译的不错

LeoG0816

顶 哦! 10个月前

1人顶

Javascript的基于原型的继承和基于类的继承在一些细微的方面是不同的。通常这不是问题，但是差别往往会在使用$scope时出现。在AngularJS中每一个$scope都从它的父$scope继承过来，最高层是$rootScope。（$scope在指令中表现的有些不同，指令中的隔离作用域仅继承那些显式声明的属性。）

从父级那里分享数据对于原型继承来说并不重要。不过如果不小心的话，会遮蔽父级$scope的属性。

我们想在导航栏上呈现一个用户名，然后进入登陆表单。

```
<div ng-controller="navCtrl">
   <span>{{user}}</span>
   <div ng-controller="loginCtrl">
        <span>{{user}}</span>
```

```
        <input ng-model="user"></input>
    </div></div>
```

翻译的不错

LeoG0816
顶 哦!10个月前

1人顶

考你下：当用户在设置了ngModel的文本框中输入了值，哪个模板会被更新？是navCtrl，loginCtrl还是两者？

如果你选loginCtrl，那么你可能对原型继承的机理比较了解了。当寻找字面值时，原型链并没有被涉及。如果navCtrl要被更新的话，那么查找原型链是必要的。当一个值时对象的时候就会发生这些。（记住在Javascript中，函数、数组合对象都算作对象）

所以想要获得期望的效果就需要在navCtrl上创建一个对象可以被loginCtrl引用。

```
<div ng-controller="navCtrl">
   <span>{{user.name}}</span>
   <div ng-controller="loginCtrl">
        <span>{{user.name}}</span>
        <input ng-model="user.name"></input>
   </div></div>
```

现在既然user是一个对象了，原型链会被考虑进去，navCtrl的模板和$scope也会随着loginCtrl更新。

这可能看上去像一个设计好的例子，但当涉及到像ngRepeat那样会创建子$scope的时候问题就会出现。

翻译的不错

LeoG0816
顶 哦!9个月前

1人顶

9 手工测试

虽然测试驱动开发可能不是每一个开发者都喜欢的开发方式，不过每次开发者去检查他们的代码是否工作或开始砸东西时，他们正在做手工测试。

没有理由不去测试一个AngularJS应用。AngularJS从一开始就是被设计地易于测试的。依赖注入和ngMock模块就是证据。核心团队开发了一些工具来讲测试带到另一个级别。

9.1 Protractor

单元测试是一组测试集的基本元素，但随着应用复杂性的提高，集成测试会引出更多实际问题。幸运地是AngularJS核心团队提供了必要的工具。

“我们构建了Protractor，一个端对端的测试运行工具，模拟用户交互，帮助你验证你的Angular应用的运行状况。”

Protractor使用Jasmine测试框架来定义测试。Protractor为不同的页面交互提供一套健壮的API。

有其他的端对端工具，不过Protractor有着自己的优势，它知道怎么和AngularJS的代码一起运行，特别是面临$digest循环的时候。

翻译的不错

LeoG0816
顶 哦!9个月前

1人顶

9.2 Karma

一旦使用Protractor写好了集成测试，测试需要被运行起来。等待测试运行特别是集成测试，会让开发者感到沮丧。AngularJS核心团队也感到了这个痛苦并开发了Karma。

Karma是一个Javascript测试运行工具，可以帮助你关闭反馈循环。Karma可以在特定的文件被修改时运行测试，它也可以在不同的浏览器上并行测试。不同的设备可以指向Karma服务器来覆盖实际场景。

翻译的不错

BreakingBad
顶 哦!10个月前

1人顶

10 jQuery的使用

jQuery 是个很不错的类库. 它将跨平台开发标准化. 在现代网页开发中具有很重要的地位. 虽然 jQuery 拥有许多强大的功能. 但是他的设计理念却与 AngularJS 大相径庭.

AngularJS 是用来开发应用框架的; jQuery 则是一个用来简化 HTML 文档对象遍历和操作, 事件处理, 动画以及 Ajax 使用的类库而已. 这是它们俩在本质上的区别. AngularJS 侧重点在于应用的架构, 而非仅仅是补充 HTML 网页的功能.

如文档所述 AngularJS 可以让你根据应用的需要对 HTML 进一步扩展. 所以, 如果想要深入的了解 AngularJS 应用开发, **就不应该再继续抱着 jQuery 的大腿**. jQuery 只会把程序员的思维方式限制在现有的 HTML 标准里头.

翻译的不错

LeoG0816

DOM操作应该出现在指令中，但这并不意味着一定要使用jQuery包装集。在使用jQuery前要考虑到一些功能AngularJS已经提供了。指令建立于相互之间，并可以创建有用的工具。

总有一天，使用jQuery库是必要的，不过从一开始就引入它无疑是一个错误。

总结

AngularJS是一个很不错的框架，并且和它的社区一起发展着。符合习惯的AngularJS仍旧是一个正在发展的概念，但希望以上这些对于规划一个AngularJS应用时会出现的陷阱可以被避免。

顶 哦! 于9个月前

1人顶

网友评论 共24条 发表评论 回页面顶部

- 开源中国匿名会员 发表于 2014-10-09 08:32 回复
  Batarang 不怎么好用。我现在希望的是谷歌的 Chrome Dev Editor 能支持 ng。
- 厄煮 发表于 2014-10-09 08:38 回复
  先马克
  中午休息的时候看
- MingunYang 发表于 2014-10-09 08:49 回复
  get
- 闲.大赋 发表于 2014-10-09 08:59 回复
  虽然有
  多年开发js的经验，但一个都看不懂，好高大尚啊
- limichange 发表于 2014-10-09 09:19 回复
  肿胀改
  成膨胀会比较好吧？
- limichange 发表于 2014-10-09 09:24 回复
  @红薯
  ，翻译区要不要加个‘踩‘的功能，别让那些机翻的人乱贴了。
- 1024 发表于 2014-10-09 09:25 回复
  ng-
  inspector for AngularJS 这个比较好用
- 红薯 发表于 2014-10-09 09:25 回复
  引用来自“耀耀”的评论
  @红薯 ，翻译区要不要加个‘踩‘的功能，别让那些机翻的人乱贴了。
  我们会审核的，我们不希望翻译里有踩的功能，如果大家觉得不好可以进入纯英文模式重新翻译
- 

hantsy 发表于 2014-10-09 09:47 回复

没仔细

看，，，第一点我觉得就在误导人。
我在项目中使用的是第一种情况方便管理，，，更重要是方便在生产环境中压缩输出。项目中只有一个主模板，templates下的所有 html模板都可以转换成 $templateCache 的 js文件，其他的 Controller，service 都是压缩成一个 js. 最终生产环境运行时，整个项目只有几个文件。

ToSun 发表于 2014-10-09 10:25 回复

看看不

同的意见也有好处，对于新手来说

开源中国匿名会员 发表于 2014-10-09 10:49 回复

> 引用来自“hantsy”的评论
> 没仔细看，，，第一点我觉得就在误导人。
> 我在项目中使用的是第一种情况方便管理，，，更重要是方便在生产环境中压缩输出。项目中只有一个主模板，templates下的所有 html模板都可以转换成 $templateCache 的 js文件，其他的 Controller，service 都是压缩成一个 js. 最终生产环境运行时，整个项目只有几个文件。

……压缩之后的东西是给机器看的。压缩之前的东西是给人看的。我觉得我还是愿意当人，不愿意当机器人。

军区文工团 发表于 2014-10-09 10:54 回复

> 引用来自“hantsy”的评论
> 没仔细看，，，第一点我觉得就在误导人。
> 我在项目中使用的是第一种情况方便管理，，，更重要是方便在生产环境中压缩输出。项目中只有一个主模板，templates下的所有 html模板都可以转换成 $templateCache 的 js文件，其他的 Controller，service 都是压缩成一个 js. 最终生产环境运行时，整个项目只有几个文件。
>
> 引用来自“开源中国匿名会员”的评论
> ……压缩之后的东西是给机器看的。压缩之前的东西是给人看的。我觉得我还是愿意当人，不愿意当机器人。

讚同，上線的時候是要打包壓縮的，開發環境和線上環境是不同的

大漠穷秋 发表于 2014-10-09 11:21 回复

这篇文

章里面说的内容略浅了，真心推荐大家来看我录的视频教程《AngularJS实战》，完全开源免费。
http://www.imooc.com/learn/156
国内第一个完整的AngularJS视频教程，从代码到理论以及各种基于NodeJS的前端开发工具，你真的值得拥有！内容简介如下：
第1章 快速上手
1-1 课程简介
1-2 快速上手
1-3 开发、调试、测试工具
第2章 基本概念和用法
2-1 MVC
2-2 路由、模块、依赖注入
2-3 双向数据绑定
2-4 路由 (12:39)
2-5 指令（1） (03:44)
2-6 Service与Provider（1） (09:19)
2-7 综合应用BookStore (24:25)

第3章 核心原理解析
3-1 第三章简介 (06:40)
3-2 AngularJS的启动过程（1）
3-3 Provider与Injector（1） (14:11)
3-4 指令的执行过程分析（1） (14:18)
3-5 $scope与双向数据绑定原理分析

第4章 用AngularJS开发移动APP

第5章 前端自动化测试

limichange 发表于 2014-10-09 11:54 回复

等

angular到2.0再考虑用来开发APP

limichange 发表于 2014-10-09 11:54 原文下 回复

面也有很多的人在吐槽文章

kingdelee 发表于 2014-10-09 12:41 回复

引用来自“大漠穷秋”的评论

这篇文章里面说的内容略浅了，真心推荐大家来看我录的视频教程《AngularJS实战》，完全开源免费。
http://www.imooc.com/learn/156
国内第一个完整的AngularJS视频教程，从代码到理论以及各种基于NodeJS的前端开发工具，你真的值得拥有！内容简介如下：
第1章 快速上手
1-1 课程简介
1-2 快速上手
1-3 开发、调试、测试工具
第2章 基本概念和用法
2-1 MVC
2-2 路由、模块、依赖注入
2-3 双向数据绑定
2-4 路由 (12:39)
2-5 指令（1） (03:44)
2-6 Service与Provider（1） (09:19)
2-7 综合应用BookStore (24:25)

第3章 核心原理解析
3-1 第三章简介 (06:40)
3-2 AngularJS的启动过程（1）
3-3 Provider与Injector（1） (14:11)
3-4 指令的执行过程分析（1） (14:18)
3-5 $scope与双向数据绑定原理分析

第4章 用AngularJS开发移动APP

第5章 前端自动化测试

大神超赞~~~我去学习学习^_^

开源狂人 发表于 2014-10-09 13:24 回复

引用来自“耀耀”的评论

肿胀改成膨胀会比较好吧？

膨胀要是改成勃Q会更^_^

tntest 发表于 2014-10-09 13:39 回复

引用来自“大漠穷秋”的评论

这篇文章里面说的内容略浅了，真心推荐大家来看我录的视频教程《AngularJS实战》，完全开源免费。
http://www.imooc.com/learn/156
国内第一个完整的AngularJS视频教程，从代码到理论以及各种基于NodeJS的前端开发工具，你真的值得拥有！内容简介如下：
第1章 快速上手
1-1 课程简介
1-2 快速上手
1-3 开发、调试、测试工具
第2章 基本概念和用法
2-1 MVC
2-2 路由、模块、依赖注入
2-3 双向数据绑定
2-4 路由 (12:39)
2-5 指令（1） (03:44)
2-6 Service与Provider（1） (09:19)
2-7 综合应用BookStore (24:25)

第3章 核心原理解析
3-1 第三章简介 (06:40)
3-2 AngularJS的启动过程（1）
3-3 Provider与Injector（1） (14:11)
3-4 指令的执行过程分析（1） (14:18)
3-5 $scope与双向数据绑定原理分析

第4章 用AngularJS开发移动APP

第5章 前端自动化测试

看看去

hythht 发表于 2014-10-09 14:46 学习 回复

了，谢谢大神~~

半夏羽涅 来自 Android 发表于 2014-10-09 20:48 回复

new Thread()

发表评论

回评论顶部 | 回页面顶部