Angular 在洋葱圈的实践与思考

—— 游志军

洋葱圈 · 分享体育快乐

2014 年 6 月发布第一个正式App版本
2015 年 4 月发布第一个正式Web版本

我们致力于缔造一个趣味与品质并举的体育社区，帮助人们更好地理解体育文化、分享体育快乐。

sponia.com

发布帖子 发布图片

软软 16:02

Sam_ 赞了帖子

逮噶猴！参加活动云云送tee！

猜一哈哪个是我，猜对（夸我）送个人珍藏葱圈Tee 好，完善下规则，避免被类似迪爷的人钻空子，赞本帖猜对，赞数最多讲的我又很高兴的就送！ 截至7月7号！ 葱圈白tee L码一件儿！ 男女皆可，顺丰包邮！... 显示全部


♡ 15 💬 13 •••

Muzi 12:57

SakuraFall 赞了帖子

抽我抽我抽我>.<

Aj1 当年这双红黑13年再出的时候超难买！ 当时朋友去通宵排队 排了2家店.最后才顺手帮我带了双童鞋版的😂 😂 😂 New Balance 576 当年韩国入手的.全白!主要比574更加修身! 好呐... 显示全部


我的草稿

我的收藏

我的订阅

下载洋葱圈客户端

支持 iOS 7 及 Android 4.0 以上系统

推荐用户 查看更多

均荣丶
哪怕漂浮，却不沉沦
79 粉丝 | 18 发布 • 关注

RM小白蛾 - 梅吹亚种
十冠战舰—¡Hala Madrid!
228 粉丝 | 51 发布 • 关注

靳天羽
懒熊体育特约作者 知乎：靳天羽T...
173 粉丝 | 16 发布 • 取关

热门圈子 查看更多

童鞋们好

今天主要谈谈:

1. 为什么选择 Angular

2. 一些小Tip

3. 怎样更好的组织项目结构

4. 结合 ES6 和 JSPM 构建更好的应用

Why Angular?

数据中间页


足球
球队
球员
赛程
排名
全部球队
热门联赛
世界杯
英超联赛
西甲联赛
德甲联赛
意甲联赛
法甲联赛
中超联赛
欧洲冠军联赛
欧洲联赛
亚洲冠军联赛
国际
欧洲
亚洲
南美洲
中超联赛 / 广州恒大 VS 广州富力
基本信息
广州恒大
赛事 中超联赛
日期 2015-05-23
比赛周
开球 20:00:00
半场比分 0 : 1
终场比分 2 : 2
广州富力
R&F
详细赛况
全部
球队
球员
事件
时间
广州富力 米切尔 点球 86'
广州富力 卢琳 ↑ 汪嵩 ↓ 轮换 84'
广州恒大 高拉特 郑龙 进球 79'
广州富力 叶楚贵 ↑ 阿隆 ↓ 轮换 79'
广州恒大 郑智 黄牌 76'
广州恒大 郜林 点球 74'
广州富力 阿隆 黄牌 72'
广州恒大 高拉特 ↑ 赵旭日 ↓ 轮换 71'
广州恒大 郑智 ↑ 弋腾 ↓ 轮换 71'
广州富力 闵俊麟 ↑ 常飞亚 ↓ 轮换 68'
广州恒大 梅方 黄牌 62'
广州恒大 王军辉 ↑ 邹正 ↓ 轮换 52'
广州恒大 张佳祺 黄牌 42'

我们所期望的:

- 前后端分离
- 应用逻辑与DOM解耦
- 大大提高生产力

为什么使用Angular 1.X?

1. 双向数据绑定

2. 强大的指令

3. 依赖注入

4. 代码结构

Angular1.X也存在一些问题:

1. 数据绑定过多性能问题

2. 概念繁多，学习路线曲折

3. Angular2.0 重写，不兼容以前版本

My Feelings About **AngularJS** Over Time

为什么还要继续使用Angular?

1. 世界上没有完美的框架

2. 采用更好的实践来避免Angular不好的东西

3. 谷歌和Angular正在变得越来越好

	version	time (ms)	garbage (kb)
	1.2.0	6,502.19	169,263.81
	1.3.0	1,468.89	44,915.07
DOM manipulation		**4.4x faster**	**73% less**
	1.2.0	159.42	4,389.48
	1.3.0	45.08	590.91
digest		**3.5x faster**	**87% less**

based on largetable benchmark with ngBind, 1.3.0 vs 1.2.0

Visit source

一些小Tip

Tip 1: 使用controllerAs

```
<div ng-controller="AppCtrl as vm">
   {{ vm.title }}
</div>

angular.module('app', [])
    .controller("AppCtrl", AppCtrl);

function AppCtrl() {
    this.title = "Hello World";
}
```

- 创建隔离上下文
- 避免$scope继承的带来的影响

```
<div ng-controller="ParentCtrl">
   {{ title }}
   <div ng-controller="ChildCtrl">
       {{ title }}
    </div>
</div>
```

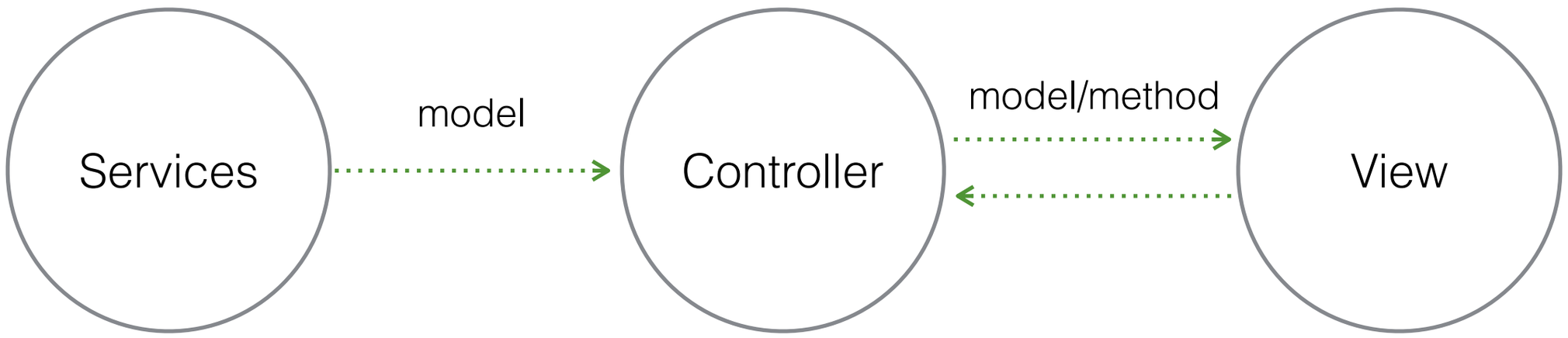
难以追踪数据来源

Tip 2: 精简你的Controller

不要将业务逻辑代码处理包含在controller里面

Why?

- Controller生命周期与View同在
- 不可复用（与View绑在一起）
- 不可共享



服务

- 获取和装饰数据
- 共享数据给Controller

控制器

- 暴露数据与方法给View

避免这样做

```
angular.module('app', [])
    .controller("TodoCtrl", TodoCtrl);

function TodoCtrl() {
    var vm = this;
    vm.todos = [];
    vm.addTodo = addTodo;
    vm.removeFromTodos = removeFromTodos;

    function addTodo(item) {
        // Business logic inside controller
        vm.todos.push(item);
    }

    function removeFromTodos(item) {
        // Business logic inside controller
        if (vm.todos.indexOf(item) > -1) {
            vm.todos.splice(vm.todos.indexOf(item), 1);
        }
    }
}
```

业务逻辑处理放到Service

```
angular
    .module('app', [])
    .factory('Todo', Todo);

/* @ngInject */
function Todo() {
    var service = {
        todos: [],
        addTodo: addTodo,
        removeFromTodos: removeFromTodos
    };
    return service;

    function addTodo(item) {
        this.todos.push(item);
    }

    function removeFromTodos(item) {
        if (this.todos.indexOf(item) > -1) {
            this.todos.splice(this.todos.indexOf(item), 1);
        }
    }
}
```

精简后的Controller

```
angular.module('app', [])
    .controller("TodoCtrl", TodoCtrl);

/* @ngInject */
function TodoCtrl(Todo) {
    var vm = this;
    vm.todos = Todo.todos;
    vm.addTodo = addTodo;
    vm.removeFromTodos = removeFromTodos;

    function addTodo(item) {
        Todo.addTodo(item);
    }

    function removeFromTodos(item) {
        Todo.removeFromTodos(item);
    }
}
```

Tip 3: 路由使用Resolve

激活Controller前加载完所需数据

仅仅数据不同

共用Controller与模板

```
/* @ngInject */
function config($stateProvider) {

    var groupHot = {
        url: '/hot',
        name: 'group.hot',
        templateUrl: 'app/group/partials/group-list.html',
        resolve: { /* @ngInject */
            groupList: function(Group) {
                return Group.getRecommendList();
            }
        },
        controller: 'GroupListCtrl as vm'
    };

    var groupSoccer = {
        url: '/soccer',
        name: 'group.soccer',
        templateUrl: 'app/group/partials/group-list.html',
        resolve: { /* @ngInject */
            groupList: function(Group) {
                return Group.getSoccerList();
            }
        },
        controller: 'GroupListCtrl as vm'
    };

    $stateProvider.state(groupHot);
    $stateProvider.state(groupSoccer);
}
```

数据注入 Controller

```
angular
    .module('app.group')
    .controller('GroupListCtrl', GroupListCtrl);

/* @ngInject */
function GroupListCtrl(groupList, Loadmore) {
    /*jshint validthis: true */
    var vm = this;
    vm.groupList = groupList;
    vm.loadmore = loadmore;

    function loadmore() {
        Loadmore.call(vm.groupList);
    }
}
```

Tip 4: 尽量使用单向数据绑定

For Performance

AngularJS 1.3+ use “one time binding”

```
<p>My name is {{::name}}</p>
```

AngularJS < 1.3 use bindonce.js

```
<p>My name is <span bo-text="name"></span></p>
```

https://github.com/Pasvaz/bindonce

怎样更好的组织项目结构?

按文件类型

```
app/
  controllers/
  services/
  partials/
```

按功能模块

```
app/
  group/
  post/
  home/
```

1、按文件类型划分:

```
src/
|
|- app/
|   |- app.js
|   |- controllers/
|     |- sidebar.js
|     |- dashboard.js
|   |- services/
|     |- util.js
|     |- post.js
|   |- directive/
|   |- partials/
|     |- sidebar.html
|     |- dashboard.html
|   |- ……
|
|- assets/
|   |- styles/
|     |- main.scss
|     |- base/
|     |- components/
|     |- helper/
|     |- pages/
|   |- images/
|   |- font/
|
|- index.html
bower_components/
node_modules/
gulpfile.js
```

优点:

- 结构清晰

缺点:

- 相关文件对应困难
- 项目大的情况下寻找文件麻烦

2、按功能模块划分:

```
src/
|
|- app/
|   |- app.module.js
|   |- group/
|     |- group.module.js
|     |— group.route.js
|     |— group-home.controller.js
|     |— ooGroupCard.js
|     |- partials/
|         |- group-home.html
|         |- group-card.html
|   |- post/
|     |- post.module.js
|     |- post.route.js
|   |- ……
|- commom/
|   |- filter/
|   |- factories/
|
|- assets/
|   |- styles/
|   |- images/
|   |- font/
|
|- index.html
bower_components/
node_modules/
gulpfile.js
```

优点:

- 能快速找到对应文件
- 以功能模块组织项目
- 易于扩展

缺点:

- 模块足够大时一级目录文件多

2、按功能模块划分:

```
src/
|
|- app/
|   |- app.module.js
|   |- group/
|     |- group.module.js
|     |- group.route.js
|     |- group-home.controller.js
|     |- ooGroupCard.js
|     |- partials/
|         |- group-home.html
|         |- group-card.html
|   |- post/
|     |- post.module.js
|     |- post.route.js
|   |- ……
|- commom/
|   |- filter/
|   |- factories/
|
|- assets/
|   |- styles/
|   |- images/
|   |- font/
|
|- index.html
bower_components/
node_modules/
gulpfile.js
```

```
// app.module.js

(function() {
    'use strict';

    angular
        .module('app', [
            'app.group',
            'app.post'
        ]);
})();
```

```
// group.modlue.js

(function() {
    'use strict';

    angular
        .module('app.group', []);

})();
```

保持一致性

```
src/
|
|- app/
|   |- app.module.js
|   |- {{module_name}}/
|     |- {{module_name}}.module.js
|     |— {{module_name}}.route.js
|     |— {{module_name}}-{{sub_name}}.controller.js
|     |— oo{{directive_name}}.js
|     |- partials/
|         |-{{module_name}}-{{sub_name}}.html
|         |-{{directive_name}}.html
|   |- group/
|     |- group.module.js
|     |- group.route.js
|     |— group-home.controller.js
|     |— ooGroupCard.js
|     |- partials/
|         |- group-home.html
|         |- group-card.html
|   |- ……
|- commom/
|   |- filter/
|   |- factories/
|
|- assets/
```

困扰我们的几个问题

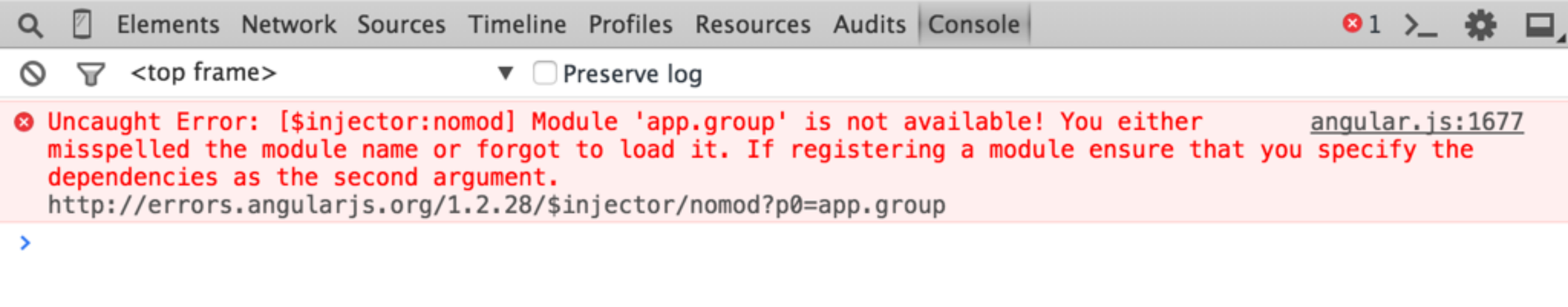
1、index.html引用一堆js文件

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Hello Angular</title>
</head>
<body>
  <script src="app/app.module.js"></script>
  <script src="app/post/post.module.js"></script>
  <script src="app/group/group.module.js"></script>
  <script src="app/home/home.module.js"></script>
  <script src="app/core/core.module.js"></script>
  <script src="app/core/config.js"></script>
  <script src="app/core/constants.js"></script>
  <script src="app/core/run.js"></script>
  <script src="app/app.controller.js"></script>
  <script src="app/group/group-detail.controller.js"></script>
  <script src="app/group/group-explore.controller.js"></script>
  <script src="app/group/ooGroupCard.js"></script>
  <script src="app/post/post.controller.js"></script>
  .....
</body>
</html>
```

```
// gulpfile.js

var gulp = require('gulp');
var config = require('./config')();
var $ = require('gulp-load-plugins')();


gulp.task('inject', function() {
    log('inject the app js into the html');
    return gulp
        .src(config.index)
        .pipe($.inject(gulp.src(config.js), {
            read: false,
            starttag: '<!-- inject:app -->',
            addRootSlash: false,
            relative: true
        }))
        .pipe(gulp.dest(config.client));
});
```

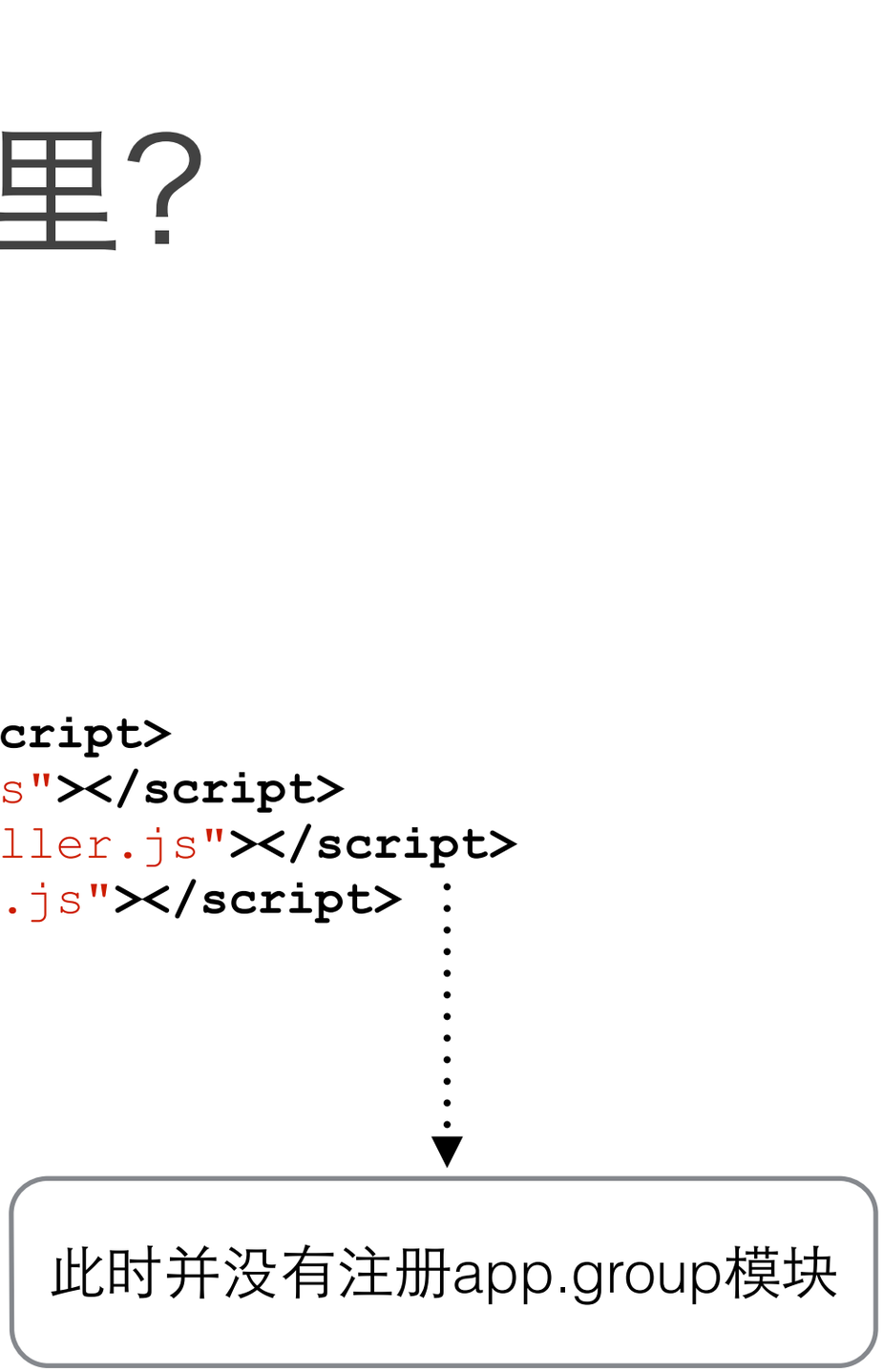
```
// index.html

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Hello Angular</title>
</head>
<body>
  <!-- inject:app -->
  <!-- endinject -->
</body>
</html>
```

但有时你可能会遇到这种情况



问题出在哪里?

```
// index.html

<body>
  <!-- inject:app -->
  <script src="app/app.module.js"></script>
  <script src="app/post/post.module.js"></script>
  <script src="app/group/group.controller.js"></script>
  <script src="app/group/group.module.js"></script> :
  <!-- endinject -->
</body>
```



怎样解决这个问题?

```
// gulpfile.js

var gulp = require('gulp');
var config = require('./config')();
var $ = require('gulp-load-plugins')();


gulp.task('inject', function() {
    log('inject the app js into the html');
    return gulp
        .src(config.index)
        .pipe($.inject(gulp.src(config.js), {
            read: false,
            starttag: '<!-- inject:app -->',
            addRootSlash: false,
            relative: true
        }))
        .pipe(gulp.dest(config.client));
});
```

```
// config.js

module.exports = function() {
    var client = 'src/';
    var clientApp = client + 'app/';
    var clientCommon = client + 'common/';

    var config = {
        index: client + 'index.html',
        js: [
            clientApp + '**/*.module.js',
            clientCommon + '**/*.module.js',
            clientApp + '**/*.js',
            clientCommon + '**/*.js',
            '!' + clientApp + '**/*.spec.js'
        ]
    };

    return config;
};
```

优先inject模块文件

更好的解决方案?

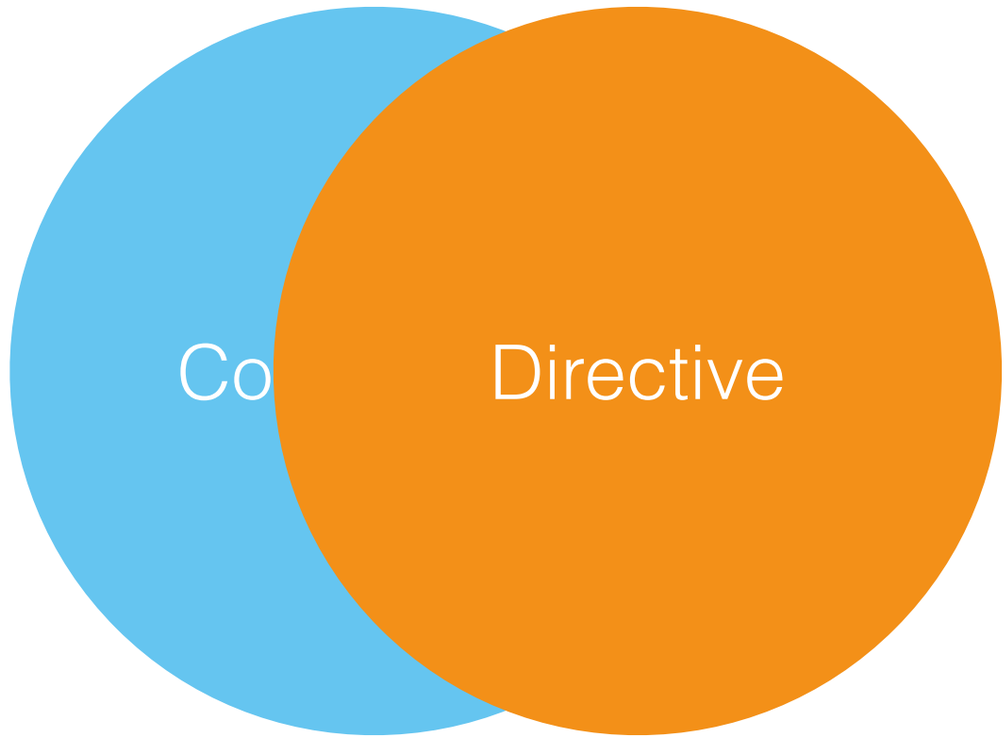
2、样式与模板分离

```
src/
|
|- app/
|   |- app.module.js
|   |- group/
|     |- group.module.js
|     |— group.route.js
|     |— group-home.controller.js
|     |— ooGroupCard.js
|     |- partials/
|         |- group-home.html
|         |- group-card.html
|
|- assets/
|   |- styles/
|     |- main.scss
|     |- base/
|     |- components/
|         |-  card.scss
|     |- helper/
|     |- pages/
|         |- _group.scss
|   |- images/
|   |- font/
|
|- index.html
bower_components/
node_modules/
gulpfile.js
```

缺点:

- 文件对应麻烦
- 难以维护和修改
- 组件化困难

更好的解决方案?


Co
Directive

// home.js

使用Directive来组织App

// home.html

```
<navbar></navbar>
<main>
  <div>
    <oo-publish></oo-publish>
    <oo-timeline></oo-timeline>
  </div>
  <sidebar></sidebar>
</main>
```

```
import angular from 'angular';
import 'angular-ui-router';
import homeComponent from './home.component';

let homeModule = angular.module('home', [
    'ui.router'
])
.config(($stateProvider)=>{
    $stateProvider
        .state('home', {
            url: '/',
            template: '<home></home>'
        });
})
.directive('home', homeComponent);

export default homeModule;
```

What About ES6

ES6 环境

转换工具:

Bebel　　Traceur

模块加载:

SystemJS　　ES6 module loader

模块管理:

JSPM

jspm.io

- 基于SystemJS的Javascript包管理器

- 可从端点如npm或github加载任何格式（ES6 AMD CommomJS）的模块

- 完善的打包功能

ES6 + 

 +  + BABEL

结合ES6和JSPM

```
src/
|- app/
|   |- common/
|     |- navbar/
|     |- sidebar/
|   |- components/
|     |- home/
|        |- home.js
|        |- home.component.js
|        |- home.controller.js
|        |- home.spec.js
|        |- home.html
|        |- _home.scss
|   |- app.component.js
|   |- app.js
|   |- app.html
|   |- _app.scss
|
|- index.html
|- jspm_packages
|    |- github/
|    |- npm/
|    |- system.js
|    |- ...
|- node_modules/
|- jspm.config.js
|- gulpfile.js
|- package.json
|- karma.conf.js
```

no more js file in index.html

```
// index.html

<body>
  <app></app>
  <!-- build:js -->
  <script src="jspm_packages/system.js"></script>
  <script src="jspm.config.js"></script>
  <script>
    System.import('app/app');
  </script>
  <!-- endbuild -->
</body>
```

```
// jspm.config.js

System.config({
  "baseURL": "./",
  "transpiler": "babel",
  "paths": {
    "*": "*.js",
    "github:*": "jspm_packages/github/*.js",
    "npm:*": "jspm_packages/npm/*.js"
  },
  "map": {
    "angular": "github:angular/bower-angular@1.4.1",
   "angular-ui-router": "github:angular-ui/ui-
router@0.2.15",
    "babel": "npm:babel-core@5.5.5",
  }
});
```

结合ES6和JSPM

```
src/
|- app/
|   |- common/
|     |- navbar/
|     |- common.js
|   |- components/
|     |- home/
|        |- home.js
|        |- home.component.js
|        |- home.controller.js
|        |- home.spec.js
|        |- home.html
|        |- _home.scss
|     |- components.js
|   |- app.component.js
|   |- app.js
|   |- app.html
|   |- _app.scss
|
|- index.html
|- jspm_packages
|    |- github/
|    |- npm/
|    |- system.js
|    |- ...
|- node_modules/
|- jspm.config.js
|- gulpfile.js
|- package.json
```

```
// app.js

import angular from 'angular';
import 'angular-ui-router';
import Common from './common/common';
import Components from './components/components';
import AppComponent from './app.component';

let appModule = angular.module('app', [
    'ui.router',
    Common.name,
    Components.name
])
.directive('app', AppComponent);

angular.element(document).ready(()=> {
  angular.bootstrap(document, [appModule.name]), {
    strictDi: true
  }
});

export default appModule;
```

Advantage of use ES6 and JSPM

Module System

Import & Export pieces of code

```
// home.js

import angular from 'angular';
import 'angular-ui-router';
import homeComponent from './home.component';

let homeModule = angular.module('home', [
        'ui.router'
    ])
    .config(($stateProvider, $urlRouterProvider) => {
        $stateProvider
            .state('home', {
                url: '/',
                template: '<home></home>'
            });
    })
    .directive('home', homeComponent);

export default homeModule;
```

Classes

Sugar Over Prototypical Inheritance

```
// parent.js

class Parent {
    constructor(name) {
        this.name = name;
    }
    getName() {
        return this.name;
    }
}

export default Parent;
```

```
// child.js

import Parent from './parent';

class Child extends Parent {
    constructor(name, age) {
        super(name);
        this.age = age;
    }
    getAge() {
        return this.age;
    }
}

export default Child;
```

Classes

Working On Controller

注意依赖注入问题

```
// home.controller.js

class HomeController {
    constructor($http, otherService) {
        this.$http = $http;
    }
    getData() {
        this.$http.get(); // works
        otherService.doSomething(); // throws error
    }
}

HomeController.$inject = ['$http', 'otherService'];

export default HomeController;
```

Arrow Function

less keystrokes, keep context.

```
// user.controller.js

class UserController {
    constructor(User) {
        this.User = User;
        this.name = 'zeejune'
    }
    getData() {
        this.User.get().then(respond => {
            // this in here === this out there
            this.name = respond.name;
        });
    }
}

UserController.$inject = ['User'];

export default UserController;
```

Template Strings

Multiline, Interpolatable Strings

```
// home.component.js

import controller from './home.controller';
import template from './home.html!text';    // JSPM will handel this

let homeComponent = function(){
    return {
        restrict: 'E',
        template,          // no more templateUrl and $templateCache
        scope: {},
        bindToController: true,
        controller,
        controllerAs: 'vm'
    };
};

export default homeComponent;
```

拥抱ES6

Last But most Important

加入我们

jobs@sponia.com

Thank You!