

# 后ANGULAR时代二三事

@民工精髓V

# 最火的前端组件框架

- Angular, 2013年下半年到2014年下半年
- React, 2014年下半年开始
- Polymer

# 为什么ANGULAR要改变

- 对Web标准的迎合 (Web Components, ES6)
- 解决自己的一些问题
  - 性能 (脏检测)
  - 模块机制
  - 部分概念设计过于复杂 (directive)

# 相同理念的后续框架

- Angular 2
- Aurelia
- 两者对比：<http://blog.durandal.io/2015/05/20/porting-an-angular-2-0-app-to-aurelia/>

# ES6与转译语言

- 使用更新的语法，一般是有好处的
- ES6提供了大量的语法糖，可以使代码更精炼
- 也可以尝试使用TypeScript等转译语言



# ES6

- 部分浏览器支持了一些主要的ES6特性
- 在生产过程中使用ES6，一般会用Babel构建成ES5代码
- 推荐阅读：使用ES6进行开发的思考 <http://otakustay.com/es6-develop-overview/>

# TYPESCRIPT

- TypeScript提供了更好的可靠性保证
- Angular 2 和Aurelia都支持使用它编写业务代码
- 在这种场景下使用TS编写代码，更有利于提高生产力
- 工程管控能力更强

# 组件与路由

- ng-route, 不支持子路由, 集中配置
- ui-router, 支持子路由, 集中配置
- ng-new-router, 支持子路由, 动态配置
- 把路由的配置下放到组件内部, 可以让组件的使用更灵活



# 组件生命周期

- 创建之前
- 创建
- 销毁之前
- 销毁

# 我们需要怎样的组件？

- Web Components不能满足所有的组件化需求
- 结构型（Custom Element），作为独立功能被引入
  - `<div><my-calendar></my-calendar></div>`
- 行为型（Custom Attribute），右键菜单，闪烁，作为附加功能被引入
  - `<div blink>test</div>`

# 理想中的组件化





# 现实中的组件化



# 组件化与分层

- 组件由模板与逻辑构成，两者分离
- 可以整体复用，也可以只复用一个部分（主要是逻辑）



# MVVM

- 在1.2之前，Angular不算MVVM
- 1.2增加了controller as语法，无需注入控制器
- Angular的model是POJO
- 在复杂场景下，model最好有类型定义，自身带有一些方法，不要都放在vm上
- model应当与store视为一体，在数据的共享，缓存，防冲突，防脏等方面综合考虑

# 代码迁移

- 代码分层做得越好，迁移代价越小
- 逻辑层比UI层容易迁移，模型监控的部分需要重新考虑
- 模板可使用规则替换
- 指令（directive）需重新实现为自定义元素或自定义属性
- 变更检测的部分需要重新实现

# 变更检测

- 脏检测
- 存取器
- observe

# REACT与ANGULAR

- 核心理念都是组件化
- React比较精细
- Angular比较粗放
- 两者都是面向中小型Web系统，业界至今没有特别好的面向大型Web应用系统的框架



# ANGULAR的前景在哪里

- 大型Web应用，比如CRM等企业软件
- 集成需求多
- 模型复杂



“I think we agree, the past is over.”

– George W. Bush

Q & A

本主题全文地址如下：

<https://github.com/xufei/blog/issues/21>