
前端工程化不完全装逼指南
分享人：郭永峰

> 如今很多企业和团队尚未重视前端工程，又或是这个概念在很多人眼里还只是“构建工具”的代名词。

前端工程化不是一个庞大而严肃的噱头，它是和你敲下的每一行代码都息息相关的实践。

思考现状

多样

- 打开手机，查看你的手机屏幕，你很习惯的点击打开你想用的APP；
- 打开微信，去浏览你好奇的文章，把玩你喜欢的游戏；
- 你慵懒的坐在沙发上，手指在你笔记本的触摸屏上轻轻滑动，你习惯性的打开淘宝京东购物，或是去浏览新闻去灌水聊天。

这都是工程师们一行行代码敲出来的，在人们享受互联网带来的便捷的同时，我们这群码农也在享受开发带来的酸爽。

迷茫

1. 无论是什么样的产品，都有同样苦逼的程序猿同学在辛苦的搬砖。
2. 他们虽然在写着不同逻辑的代码，却都遇到相同情况的技术问题。

越来越复杂的web应用，如何规划好逻辑的调用、如何管理一堆的CSS代码、如何组织项目目录、如何团队协作、如何上线部署、如何模块化的实现代码复用.....

破局

尽管遇到的问题很多，但是，有什么事情可以难住机智如你的大脑。见招拆招，东拼西凑，在处理问题的过程中慢慢形成自己团队的方案和风格。如果你是一位有经验的前端工程师，相信你也在自己的团队里面走过了下面这些路。

前端面临的挑战

1. 技术选型

前端技术选型可真是件有趣的事情，形形色色的类库和框架的挑选，简直比相亲选姑娘还让人纠结。经常听到这样的声音，angular好还是react好，用requirejs合适还是seajs合适，选bootstrap好看还是ionic好看....根据特定项目的特定进行相应的技术选型，是前端团队必然面临的一个问题。

2. 项目构建

虽然谈到项目构建，但依然还是很多前端同学不知道Grunt或是Gulp为何物，他们依然手写css、html、js，如果要压缩代码，则百度一下在线压缩，找个工具copy代码压缩后再copy回来，如果要加版本戳，则在文件后面加上类似 `?t=2015423432` 等标识，如果需要异构less或sass，他们依然会去找个其他独立的工具来做这个事情。没错，依然会有很多的前端团队在如此工程化程度低的环境工具，也许跟公司的技术氛围有关吧。

但至少现在，市面上有很优秀的构建工具来帮我们做这些事情，比如grunt、gulp、webpack、fis、jdf、coolie等等，用它们来做代码的解析、压缩、校验、合并打包等自动化工作，我们可以很轻易的基于这些工具去搭建一套我们的前端工作流。但这只是前端工程化的一部分，我们的路还很长。

3. 模块化开发

该和一个文件几千几万行代码的时代说再见了，我们需要模块化的开发。既然web应用很复杂，那我们就应当将复杂的事情简单化，从纷乱的逻辑中抽丝剥茧般找到清晰的逻辑规划，分而治之的模块化思想，是开发和维护复杂应用的基石。

- 这里提到的模块化包括JS模块化和CSS的模块化，而其中的方案相比大家都比较熟知了。如，JS模块化方案就有AMD/CommonJS/UMD/ES6 Modlue等，与模块方案对应的框架和工具也很多；
- CSS模块化开发基本都是通过less、sass、stylus等预处理器来实现的。

实现了模块化的开发，对于团队而言，火力已上升一个数量级，至少不用再为以前杂糅在一起的代码发愁。

虽然历经了磨难，我们的前端团队经历以上几步的洗礼，已然能够Hold住公司复杂业务的需求，但是，码农的世界并未就此清净，否则，寂静的深夜也不会有那么多闪闪屏幕前极速敲击的键盘声响。自动化项目构建、模块化开发以及合理的技术选型可以满足基本业务开发，但前端面临的工程问题依然还很严峻！！！

- 如何进行多团队跨部门的进行大规模开发，如何统一开发规范
- 构建高性能的web应用，我们还有很多事情要做：
    - CDN部署
    - 缓存控制与复用
    - 文件指纹
    - 按需加载、同步/异步加载
    - 合并请求
    - ....

要做到大幅提升前端开发效率，并且兼顾web性能优化，那么首先需要考虑如何攻破 **组件化开发** 和 **静态资源管理** 两大壁垒，很显然，这已非难事，但我们依旧需要前行。

4. 组件化开发

在这React燥热的年代，相信你对组件化的说法早已是耳熟能详了。组件的概念虽很普遍，但是大家对组件的理解却各有不同，而这里说的组件指UI组件。实现组件化目前也有很多方案了，@xufei在2015.03时写的2015前端组件化框架之路 这篇文章对组件化的分析是非常深刻的。。

同时，@fouber对组件化理念的解读我也比较赞同：

- 页面的每个独立可视区域都可为一个组件
- 每个组件对应独立目录，组件资源就近维护
- 组件可自由组合，页面组件可自由删替

这也是React对组件的定义。对应的，组件化实现的方案也比较丰富：

- 通过扩展模板语法及结合工具框架实现资源重组的组件化方案
- React结合JSX语法的组件化方案
- Polymer && Web Components方案
- Angular
- Vuejs

对于组件化的实践，美团点评的 前端组件化开发实践 这篇文章讲的很不错。

5. 静态资源管理

对于前端静态资源的管理，自第一点起我们就需要面临，无论你的项目前端工程化程度如何，无论前端技术如何的演变，静态资源增量加载方案的探索都将是重中之重，是检验基础架构优劣的试金石。

资源管理的问题，说容易也容易，把html、js、css、images等资源往服务器上一扔，用户可以正常访问页面，完美，收工回家睡觉。但说复杂也复杂，比如说FB网站的建设，整站一万多个静态资源，每个资源翻译成100种语言，光这都足够让你睡不着觉了...

1. 百度FIS团队通过对FB系统分析得出的资源管理的启示让人印象深刻：

   静态资源管理系统 = 资源表 + 资源加载框架

2. 基于此再对应的思考实践：

   一个通用的资源表生成工具 + 基于表的资源加载框架

3. 基于工具和框架的性能优化方案：

   加载相关的按需加载、延迟加载、预加载、请求合并等策略；有缓存相关的浏览器缓存利用，缓存更新、缓存共享、非覆盖式发布等方案；BigRender、BigPipe、Quickling、PageCache等技术.

前端技术元素归纳

> 如果我跟你说，以上的内容都是铺垫，现在才刚刚暖场，你会不会拿板砖拍我，哈哈。

从以上提及的前端面临的挑战我们可以归纳出一个web前端团队大概会遇到的技术元素

1. 开发规范

这里的开发规范包括开发、部署的目录规范，编码规范等，团队合理的规范可以极大的提升开发效率，帮助他们快速定位问题。

2. 模块化开发

针对js、css，以功能或业务为单元组织代码。js方面解决独立作用域、依赖管理、api暴露、按需加载与执行、安全合并等问题，css方面解决依赖管理、组件内部样式管理等问题。是提升前端开发效率的重要基础。现在流行的模块化框架有requirejs、seajs等。

3. 组件化开发以及组件仓库的建设

在模块化基础上，组件单元的资源就近开发维护，组件也需要分为通用组件、业务组件、公共组件，方便最大限度地组件管理，将一些非常通用的组件放到一个公共的地方供团队共享，方便新项目复用，这个时候我们就需要引入一个组件仓库的东西，现在流行的组件库有bower、component等。团队发展到一定规模后，组件库的需求会变得非常强烈。

4. 性能优化

这里的性能优化是指能够通过工程手段保证的性能优化点。性能优化是前端项目发展到一定阶段必须经历的过程。这部分我想强调的一点是 性能优化一定是一个工程问题和统计问题，不能用工程手段保证的性能优化是不靠谱的，优化时只考虑一个页面的首次加载，不考虑全局在宏观统计上的优化提升也是片面的。

5. 项目部署

部署按照现行业界的分工标准，虽然不是前端的工作范畴，但它对性能优化有直接的影响，包括静态资源缓存、cdn、非覆盖式发布等问题。合理的静态资源资源部署可以为前端性能带来较大的优化空间。

7. 开发工具

开发工具是指包括构建与优化工具、开发-调试-部署等流程工具，以及组件库获取、提交等相关工具，甚至运营、文档、配置发布等平台工具。前端开发形成一个连贯可持续优化的开发体系需要工具支持，工具的设计至关重要。

> 一个前端团队的技术体系形成过程，无外乎是在努力拼凑上述的各项技术元素的具体解决方案，真正能够提出解决以上所有问题的方案才可以称之为集成解决方案。FIS和JDF做到了，当然也有很多优秀的团队也做到了。

FIS3的介绍和使用

FIS3 是面向前端的工程构建工具。解决前端工程中性能优化、资源加载（异步、同步、按需、预加载、依赖管理、合并、内嵌）、模块化开发、自动化工具、开发规范、代码部署等问题。

学习资料推荐

1. 官网： http://fis.baidu.com/fis3/index.html
2. 方案： http://oak.baidu.com/
3. github： https://github.com/fex-team/fis3
4. fis3的demo示例： https://github.com/fex-team/fis3-demo

JDF的介绍和使用

- JDF为京东前端开发集成解决方案
- 目的是合理，快速和高效的解决前端开发中的工程和项目问题 核心提供了前端开发必备的基础的UI和业务组件，并集成调试，构建，布署，代码生成，文档生成，编辑器插件等一系列开发工具
- 同时提供了前端模块的下载，预览，发布

学习资料

1. github： https://github.com/putaoshu/jdf

Jello的实践

> Jello是基于服务端为java，velocity/jsp为模板的解决方案。

学习地址： http://oak.baidu.com/jello-demo/

我对Jello的实践介绍

- 对整站web应用划分业务系统
- 选择后端jsp模板语言对接spring mvc架构
- js代码遵循commonjs模块化规范
- widget组件化
- 选择less预处理器进行css模块化
- 性能优化
- shell脚本部署
- 模拟数据，前后端独立开发

端上DEMO和大家共享

https://github.com/GuoYongfeng/engineering-share-demo

Q & A

THANKS