一个完整的React项目下来后的总结

我们上个月完成一个完全基于React做的单页应用（个人认为是业界第一个基于React的业务系统）
我从去年11月接触到了React，喜欢的一发不可收拾，迅速决定用它来做我们的系统

项目背景

- 物联网业务系统和互联网结合
- 4个月开发周期，从无到上线

技术环境

- 开发工具Atom + 大量插件（eslint,gitplus,react.....) + github
- 打包 es6 + Webpack + less
- 使用了HapiJs作为模拟server，前端开发和后端彻底分离，后期集成时，使用了自己开发的代理服务器调用真正的API
- 运行库 React + Flux + React-router+ material-ui + ImmutableJS + 自己研发的控件
- 单测环境 Karma+mocha+chai+Istanbul
- 布局完全用flex

个人的一点体会

- 使用Flex布局会大大降低CSS的难度
- 使用Flux可以很好的隔离业务（我们的业务逻辑都在Store里）,大大降低了单元测试的难度
- 基于React的控件（这里说的是基本控件）是对React理解的一个体现
- 尽量把React组件（Component）做到小，做到细，也就是尽量拆分React组件
- 基于数据驱动的方式（Data-Driven）开发，理解到这一点就会发现React很神奇，可能这也是最不好理解的地方
- ImmutableJS带来了大量的好处，所有组件都使用了ShouldComponentUpdate来优化
- 实现了I18N的动态加载
- 使用了Url来保存Router信息，没有使用Hash，感觉很简洁

存在的一些问题

- Flux的循环调用问题，可能是一个弊端，但是，总是可以让我们重新思考，我们这样的流程是否完全必要，还是有逻辑不清晰的地方。我遇到过几次循环调用的问题，最终都能通过不使用setTimeout来解决
- 还是存在一些冗余的代码，个人认为还是需要引入Model层（虽然我们没有这么做），因为我们是做的数据驱动，所以，有些model内容的验证（如表单验证）可能做在Model里会好些

还需要改进的地方

- Webpack打包的size还是有些大，可能代码有些冗余
- WebSocket应该在后续加入系统，可能使用Socket.io
- CSS可能还是需要规范，主要参考BEM,OOCSS等规范
- 增加更多的动画效果
- 开发一套属于自己的组件库，逐步抛弃material-ui库

总的来说，React确实大大提升了我们的开发效率，而且并没有降低运行效率。
所以严重推荐大家使用。
现在，整个团队都很熟悉了React，后面可能会做一些开源的组件。