MVVM时代的Web控件

基于AngularJS实现

@民工精髓V

变革的时代

- YUI框架停止开发
- 界面模式多样化
- 前端MV * 框架的流行
- Web Components
- 在新的时代，控件体系面临很大的挑战

AngularJS

- AngularJS是一种比较有名的前端框架，基于MVVM模式，在合适的场景使用它，会非常省心
- 如果用AngularJS，尽可能抛弃jQuery，不要混用
- 但它周边比较贫乏，尤其是UI控件。如果使用它，怎么解决这些问题?
- 毛主席教导我们：自己动手，丰衣足食

控件的基本分类

- 容器，比如Panel，TabNavigator，Accordion
- 数据列表，比如List，DataGrid，Tree
- DOM辅助，比如ContextMenu
- 公共服务，比如Alert，Dialog
- 独立功能块，比如Pager，Calendar
- 表单增强，比如DatetimePicker，ColorPicker
- 动画
- 图表

容器

- 最简单的容器比如Panel，可以直接用HTML和CSS实现
- 稍微复杂的容器，比如TabNavigator，Accordion，其数据模型很简单，有了双向绑定之后，非常容易实现
- 简单的TabNavigator demo：http://jsbin.com/homas/6
- 甚至很多控件的模型都可以共用：http://jsbin.com/homas/8
- 容器基本没有封装成控件的价值，只要设计好DOM结构和样式即可

列表

- 列表型控件侧重于数组数据的展示和操作
- 最简单的数据列表是单列数据的展示
- 无论是横向、纵向或者其他什么形态的列表，数据模型基本都是一样的
- 借助Angular的绑定机制，控件被剥离成数据模型和界面模板，想定制各种形态的列表非常容易。
- 两种形态的列表demo：http://jsbin.com/yeciga/1

数据表格

- 数据表格跟列表大同小异，只是能同时展示多列，一般我们可以用table来组成
- 决定数据表格封装方式的因素是展示的复杂度
- 简单的展示，可以不必封装，直接用HTML模板绑定数据源来实现
- 如果这个数据表格很复杂，比如行列要拖动之类，只好封装一下

树形结构

- MVVM框架中应尽量避免深层次的数据
- Angular监控大数组或者深层结构的效率偏低
- 常见的实现树控件的方式都是递归$compile，数据量大的情况下状况不佳
- 对于大数据量的树，最好还是手动处理数据，如果想让业务方便一些，可以封装成directive，内部不使用绑定机制

DOM辅助

- 基于数据绑定，绝大部分DOM操作都可以避免
- 有一些跟鼠标事件相关的东西，不得不用jQuery那种方式处理，比如右键菜单
- 即使是这类控件，也可以再作一次分离，只把位置这种跟事件相关的东西用原始方式处理，菜单项仍然使用双向绑定

公共服务

- MVVM的理念是UI和数据模型分离
- 但是，有一些UI部件会直接插入代码流程，比如，各种对话框，操作提示，如何处理?
- AngularJS的推荐实践是把DOM操作封装在directive中，但这种情况并不适合
- 这类适合做成服务，并非所有服务层都一定不能操作DOM

独立功能块

- 在AngularJS中，几乎一切都是组件化的
- 日历或者分页这种控件，本质上跟业务功能块毫无区别，只是公用度更高
- 这类控件的设计要点是隔离自身的作用域，只在视图模型上通过事件与外界沟通

表单增强

- 有一些控件是作为增强的表单项的，比如DatetimePicker，ColorPicker
- 它们跟原生的input，select等很相似，适合直接绑定到数据模型上的字段
- 这类控件与独立功能块之间的差异在于两点：
    1. 形态是否像表单项
    2. 离数据模型的距离

动画

- jQuery体系有很多动画特效
- Angular实现动画的理念整体是配置式，所以更依赖于CSS Transition和Animation，这类适合做两个状态之间的变迁
- 如果要显示很独立的动画效果，还是要用JavaScript代码去单独封装

图表

- AngularJS的理念是声明式、双向绑定，它不仅可以跟普通HTML元素绑定，还可以绑定SVG
- 一个简单的demo：http://jsbin.com/yokik/1
- 甚至不同图表可以共享数据模型：http://xufei.github.io/ng-charts/index.html
- 通过对HTML的绑定，我们不需要jQuery了；通过对SVG的绑定，我们不需要Raphael了

其他

- 智者千虑，必有一失
- 如果遇到特别复杂，自己没法写的控件，怎么集成进来?
- 三个步骤：引入第三方库，用directive包装起来，作好懒加载

总结

- 控件的本质是不同数据模型的交互展示
- 传统控件封装了数据模型与DOM的对应关系，但是在MVVM时代，界面和逻辑完全解耦，绝大部分控件的主体代码被大幅减少，不再成其为控件
- 控件的多样化，界面的单页化，导致对DOM和CSS的规划能力要求大幅增加
- 数据模型抽取出来，UI层配置化，是使用AngularJS的关键
- 详细版：https://github.com/xufei/ng-control/issues/2

“未来是光明而美丽的”

——车尔尼雪夫斯基