前端组件化研究

屈克诚　邱喜阳

（山东协和学院 计算机学院　山东　济南　250107）

摘要：随着浏览器性能提升，新的技术和形式不断升级，前端开发工程师面对的交互和设计也日趋复杂，开发周期变长，旧有的项目维护困难，与新的功能难以兼容匹配。前端开发者也需要学习和使用传统软件开发的思路和工具。其中组件化是最早在前端范围内展开应用的开发思想。本文以组件化思想为切入点，探索前端组件化的演变。

关键词：前端；组件化

1、什么是组件

Component，中文称为组件，或者构件，使用比较广泛，它的核心意义在于复用，相对模块，对于依赖性有更高的要求。Module，中文为模块或模组。它的核心意义是分离职责，属于代码级模块化的产出。它本身是提供服务的功能逻辑，是一组具有一定内聚性代码的组合，职责明确。组件(Component)和模块(Module)又是一对容易混淆的名词，也常常被用来相互替换。从设计上来看，组件强调复用，模块强调职责(内聚、分离)，或者说组件是达到可复用要求的模块。

前端 Web 应用中的组件，是指一些设计为通用性的，用来构建较大型应用程序的软件，这些组件有多种表现形式。它可以是有 UI（用户界面）的，也可以是作为“ 服务 ”的纯逻辑代码。

2、组件化思想

组件化并不是前端所特有的，一些其他的语言或者桌面程序等，都具有组件化的先例。确切的说，只要有 UI 层的展示，就必定有可以组件化的地方。简单来说，组件就是将一段 UI 样式和其对应的功能作为独立的整体去看待，无论这个整体放在哪里去使用，它都具有一样的功能和样式，从而实现复用，这种整体化的细想就是组件化。不难看出，组件化设计就是为了增加复用性，灵活性，提高系统设计，从而提高开发效率。

3、构建组件化的意义

高内聚

我们将相关的一些功能组织在一起，把一切封装起来，而在组件的例子中，就可能是相关的功能逻辑和静态资源：JavaScript、HTML、CSS 以及图像等。这就是我们所说的内聚。这种做法将让组件更容易维护，并且这么做之后，组件的可靠性也将提高。同时，它也能让组件的功能明确，增大组件重用的可能性。

可重用

功能明确，实现清晰，API 易于理解，自然就能促进组件复用。通过构建可重用组件，我们不仅保持了 DRY（不要重复造轮子）原则，还得到了相应的好处。

可互换

一个功能明确好组件的 API 能让人轻易地更改其内部的功能实现。要是程序内部的组件是松耦合的，那事实上可以用一个组件轻易地替换另一个组件，只要遵循相同的 API/接口/约定。

可组合

基于组件的架构让组件组合成新组件更加容易。这样的设计让组件更加专注，也让其他组件中构建和暴露的功能更好利用。不论是给程序添加功能，还是用来制作完整的程序，更加复杂的功能也能如法炮制。这就是这种方法的主要好处。

4、组件化的演变

在 jQuery 比较流行的时候，大部分的前端开发应该都是十分过程式的开发：操作 DOM，发起 ajax 请求，刷新数据，局部更新页面。这样的动作反反复复，甚至在同一个项目里同样的流程也许还要重复，其实 jQuery 本身也有有自己模块化的设计，有时我们也会用到类似 jQueryUI 等不错的库来减少工作量，但这种设计，更多的应该被认为是模块化。

避免频繁操作 DOM，这时开始流行 MVC，前端开始学习后端的思想，讲业务逻辑，UI，功能，可以按照不同的文件去划分，结构清晰，设计明了，开发起来也不错。在这个基础上，又有了更加不错的 MVVM 框架，它的出现，更加简化了前端的操作，并且将前端的 UI 赋予了真实意义：你所看到的任何 UI，应该都对应其相应的 ViewModel，即你看到的 view 就是真实的数据，并且实现了双向绑定，只要 UI 改变，UI 所对应的数据也改变，反之亦然。这的确很方便，但大部分的 MVVM 框架，并没有实现组件化，或者说没有很好的实现组件化，因为 MVVM 最大的问题就是：

1.执行效率，只要数据改变，它下面所有监测数据上绑定的 UI 一般都会去更新，效率很低，如果你操作频繁，很可能调了几十万遍（有可能层次太深或者监测了太多的数据变化）。

2.由于 MVVM 一般需要严格的 ViewModel 的作用域，因此大部分情况不支持多次绑定，或者只允许绑定一个根节点做为顶层 DOM 渲染，这就给组件化带来了困难（不能独立的去绑定部分 UI）。

如果你在一个大型团队工作，或者你的企业有许多部门，你们该如何实现全局 UI 组件来跨越各个板块的界限？想象一个场景，如果你的整个公司都在使用同一段 UI 代码处理公共组件，财务工具在使用它，博客工具在使用它，在线聊天工具在使用它，且无论是在移动端，桌面端还是 web 端都能见到它，那该有多便利？无需累赘而繁复地一遍又一遍实现功能类似的表单，按钮或是列表。这是一个相对理想的设置，因为无论你在哪儿你都只需要维护一个代码库，且所有全局资源也都在同一个地方，开发者们可以方便地找到所需的代码并对其贡献。同时，共享 UI 组件同事也会给你的用户带来相容的体验，无论他在浏览或使用哪个工具，移动端或是桌面端，他的所见所感都是相一致的。注意的是同步这一概念，对于拥有很多产品的公司来说如果共享 UI，那就意味着一次 UI 升级整个公司的产品都会受其影响。这从大部分意义上来说是一件好事，但有时又会带来很多麻烦，之后会说到。同时你的组件也应当是完备的，不与任何你所在团队所使用的第三方包相冲突。

5、结语：

在一个大型前端项目中，这种组件化的抽象设计是很重要的，不仅增加了复用性提高了工作效率，从某种程度上来说也反应了程序员对业务和产品设计的理解。

参考文献

[1] MarsLUL 前端 UI 组件化的一些思考 https://zhuanlan.zhihu.com/p/25820838 2017-03-17.

[2] 喜欢特别冷的冬天下着雪 前端组件化思想 https://blog.csdn.net/kkkkkxiaofei/article/details/79384219 2018-02-27.

[3] chenchun91 前端组件化开发 https://blog.csdn.net/chenchun91/article/details/53375041 2016-11-28.

[4] KevinDMitnick 基于 Angular 的前端组件化 https://blog.csdn.net/lzch2105/article/details/81175594 2018-07-24.