写在最后

亲爱的读者，到这里本小册就要结束了，你是否从中学习到了属于你的知识呢？我们来回顾一下小册的内容吧。

Vue.js 在开发独立组件时，由于它的特殊性，无法使用 Vuex、Bus 这样的第三方插件来做组件通信，因此小册提到了 3 种组件间的通信方法，都是支持跨多级的：

1. provide / inject：由父组件通过 provide 向下提供一个数据，子组件在需要时，通过 inject 注入这个依赖，就可以直接访问父级的数据了。
2. dispatch / broadcast：组件向上派发或向下广播一个自定义事件，组件通过 $on 来监听。
3. findComponents 系列：共包含 5 个方法，通过组件的 name 选项，遍历找到对应的实例，这是组件通信的终极方案，适用于所有场景。

本册总共讲解了 7 个组件的实例：

1. 具有数据校验功能的 Form 组件，它用到了第 1 种组件通信；
2. 组合多选框 Checkbox 组件，它用到了第 2 种和第 3 种组件通信；
3. Display 组件，它利用 Vue.js 的构造器 extend 和手动挂载 $mount API；
4. 全局通知 $Alert 组件，也是利用了 $mount API，与传统组件不同的是，它基于 Vue.js 组件开发，但却是以 JavaScript 的形式调用的，并且组件会在 body 节点下渲染；
5. 表格组件 Table，典型的数据驱动型组件，使用了函数式组件（Functional Render）来自定义列模板；
6. 表格组件 Table，与上例不同的是，它的自定义列模板使用了 slot-scope；

7. 树形控件 Tree，典型的数据驱动型组件，也是典型的递归组件，其中利用 computed 做父子节点联动是精髓。

最后的拓展部分，对 Vue.js 组件的常见 API 做了详细介绍，以及常见的 Vue.js 面试题分析和对开源的一些见解。

Vue.js 组件开发，归根到底拼的是 JavaScript 的功底，Vue.js 在其中只是决定了开发模式，所以，打好 JavaScript 基础才是最重要的。

最后，祝愿亲爱的读者能在编程的道路上越走越远。

另外，如果您有关于 Vue.js 或其它前端相关的问题，可以通过 iView 社区 (https://dev.iviewui.com/issues) 付费向笔者提问，希望能对一部分人有所帮助。