开篇：Vue.js 的精髓——组件

写在前面

Vue.js，无疑是当下最火热的前端框架 *Almost*，而 Vue.js 最精髓的，正是它的组件与组件化。写一个 Vue 工程，也就是在写一个个的组件。

业务场景是千变万化的，而不变的是 Vue.js 组件开发的核心思想和使用技巧，掌握了 Vue.js 组件的各种开发模式，再复杂的业务场景也可以轻松化解。本小册则着重介绍笔者在 3 年的 Vue.js 开发及两年的 iView (https://github.com/iview/iview) 开源中积累和沉淀的对 Vue.js 组件的见解和经验。

本小册**不会**介绍 Vue.js 的基础用法，因为市面上已经沉淀了大量的相关技术资料，而且 Vue.js 的文档已经足够详细。如果您尚未接触 Vue.js 或正打算开始了解，推荐您先阅读笔者出版的《Vue.js 实战》(https://item.jd.com/12215519.html)（清华大学出版社）一书，它适合刚接触 Vue.js 的开发者。因此，本小册适合已经了解或使用过 Vue.js 的开发者。

这一节，我们先笼统地聊聊 Vue.js 组件和组件化以及本小册各章节的梳理。

组件的分类

一般来说，Vue.js 组件主要分成三类：

1. 由 vue-router 产生的每个页面，它本质上也是一个组件（.vue），主要承载当前页面的 HTML 结构，会包含数据获取、数据整理、数据可视化等常规业务。整个文件相对较大，

但一般不会有 props 选项和 自定义事件，因为它作为路由的渲染，不会被复用，因此也不会对外提供接口。

在项目开发中，我们写的大部分代码都是这类的组件（页面），协同开发时，每人维护自己的页面，很少有交集。这类组件相对是最好写的，因为主要是还原设计稿，完成需求，不需要太多模块和架构设计上的考虑。

2. 不包含业务，独立、具体功能的基础组件，比如**日期选择器**、**模态框**等。这类组件作为项目的基础控件，会被大量使用，因此组件的 API 进行过高强度的抽象，可以通过不同配置实现不同的功能。比如笔者开源的 iView，就是包含了 50 多个这样基础组件的 UI 组件库。

   每个公司都有自己的组件使用规范或组件库，但要开发和维护一套像 iView 这样的组件库，投入的人力和精力还是很重的，所以出于成本考虑，很多项目都会使用已有的开源组件库。

   独立组件的开发难度要高于第一类组件，因为它的侧重点是 API 的设计、兼容性、性能、以及复杂的功能。这类组件对 JavaScript 的编程能力有一定要求，也会包含非常多的技巧，比如在不依赖 Vuex 和 Bus（因为独立组件，无法依赖其它库）的情况下，各组件间的通信，还会涉及很多脑壳疼的逻辑，比如日期选择器要考虑不同时区、国家的日历习惯，支持多种日期格式。

   本小册也会重点介绍此类组件的各种开发模式和技巧，对应不同的模式，会带有具体的组件实战。

3. 业务组件。它不像第二类独立组件只包含某个功能，而是在业务中被多个页面复用的，它与独立组件的区别是，业务组件只在当前项目中会用到，不具有通用性，而且会包含一些业务，比如数据请求；而独立组件不含业务，在任何项目中都可以使用，功能单一，比如一个具有数据校验功能的输入框。

> 业务组件更像是介于第一类和第二类之间，在开发上也与独立组件类似，但寄托于项目，你可以使用项目中的技术栈，比如 Vuex、axios、echarts 等，所以它的开发难度相对独立组件要容易点，但也有必要考虑组件的可维护性和复用性。

小册的内容

因为本小册是围绕 Vue.js 组件展开的，所以第二节会讲解 Vue.js 组件的三个 API：prop、event、slot，当然，如果你已经开发过一些独立组件，完全可以跳过这节内容。

3 - 7 小节会介绍组件间通信的一些方法和黑科技，一部分是 Vue.js 内置的，一部分是自行实现的，在实际开发中，会非常实用。同时也利用这些方法完成了两个具体的实战案例：

1. 具有数据校验功能的表单组件 —— Form；
2. 组合多选框组件 —— CheckboxGroup & Checkbox。

本小册都会以这种核心科技 + 对应实战的形式展开。

8 - 10 小节介绍 Vue 的构造器 extend 和手动挂载组件 $mount 的用法及案例。Vue.js 除了我们正常 new Vue() 外，还可以手动挂载的，这 3 节将介绍手动挂载一个 Vue 组件的使用场景。其中涉及到两个案例：

1. 动态渲染 .vue 文件的组件 —— Display；
2. 全局通知组件 —— $Alert。

Display 组件用于将 .vue 文件渲染出来，线上的案例是 [iView Run (https://run.iviewui.com/)](https://run.iviewui.com/)，它不需要你重新编译项目，就可以渲染一个标准的 Vue.js 组件。

$Alert 是全局的通知组件，它的调用方法不同于常规组件。常规组件使用方法形如：

```
<template>
    <Alert content="通知内容" :duration="3">
</Alert>
</template>
<script>
    import Alert from '../components/alert.vue';

    export default {
        components: { Alert }
    }
</script>
```

而 $Alert 的调用更接近 JS 语法：

```
export default {
    methods: {
        showMessage () {
            this.$Alert({
                content: '通知内容',
                duration: 3
            });
        }
    }
}
```

虽然与常规 Vue 组件调用方式不同，但底层仍然由 Vue 组件构成和维护。

11 - 12 小节介绍 Render 函数与 Functional Render，并完成一个能够渲染自定义列的 Table 组件。Render 函数也是 Vue.js 组件重要的一部分，只不过在大多数业务中不常使用。本小节会介绍它的使用场景。

13 小节介绍**作用域 slot（slot-scope）**，并基于这种方法同样实现 Table 组件。slot 用的很多，但 slot-scope 在业务中并不常用，但在一些特定场景下，比如组件内部有循环体时，会非常实用。

14 - 15 小节介绍递归组件，并完成树形控件 —— Tree。

16 - 19 小节是综合拓展，会着重讲解 Vue.js 容易忽略却很重要的 API，以及对 Vue.js 面试题的详细分析。除此之外，还会总结笔者在两年的 iView 开源经历中的经验，除了技术细节外，还包括开源项目的持续性发展、推广等。

结语

三年前，我开始接触 Vue.js 框架，当时就被它的轻量、组件化和友好的 API 所吸引。与此同时，我也开源了 iView 项目。三年的磨(cǎi)砺(kēng)，沉淀了不少关于 Vue.js 组件的经验。

本小册的内容也许不会让你的技术一夜间突飞猛进，但绝对使你醍醐灌顶。

那么，请准备好一台电脑和一杯咖啡，一起来探索 Vue.js 的精髓吧。