开发指南篇3：合理划分容器组件与展示组件

上篇文章我们提到了组件的概念，组件是目前模块化、组件化开发模式中必不可少的单元形式，那么除了其概念和可复用性外，我们对它的职能划分了解多少呢?

本文将立足Vue 组件的职能来谈谈我个人对于其划分的理解，唯有了解不同类型组件的职能才能编写出可维护、低耦合的前端代码。

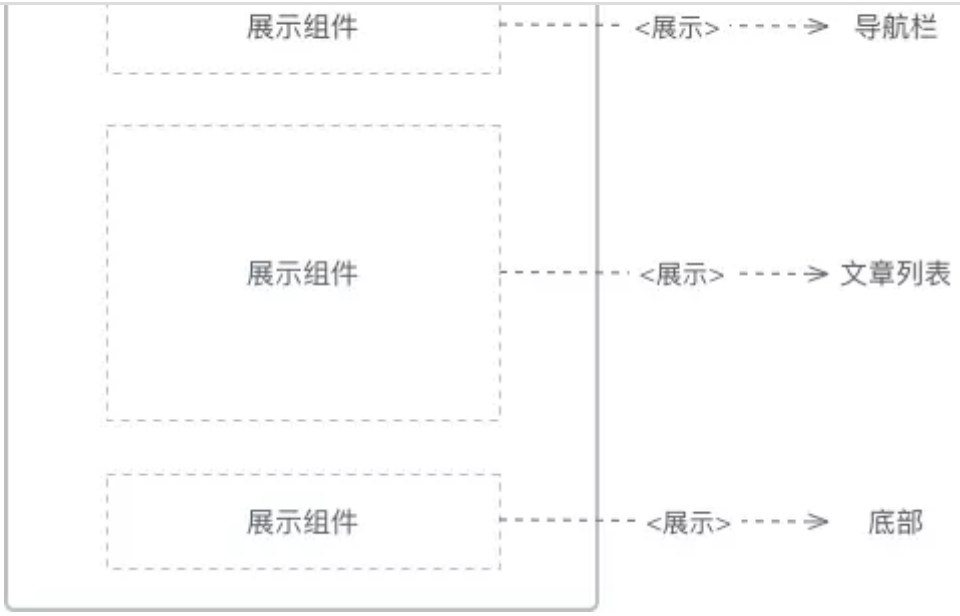
组件的职能划分

如果要将Vue 组件按照职能划分，我们可以将其分为两种类型：容器组件和展示组件。

容器组件和展示组件的概念来自于 `Redux` 文档，那么首先什么是容器组件呢？顾名思义，它是一个容器性质的组件，我们可以把它理解为最外层的父组件，也就是最顶层的组件，一般我们把它放置在 `views` 文件夹下，其功能主要用于做数据提取与实现公共逻辑，然后渲染对应的子组件。

另一类组件叫做展示组件，字面意思就是主要用于做展示的组件，其主要功能是负责接收从容器组件传输过来的数据并在页面上渲染，实现其内部独有的功能逻辑。

一个页面中容器组件与展示组件的关系如下图所示：



上图我们以博客首页为例，容器组件就是整个首页最外层的父组件，而展示组件就包含了导航栏、文章列表、底部等子组件，代码层面如下：

```html
<template>
    <div>
        <navigation @count="countFn"></navigation>
        <article :list="articleList"></article>
        <foot></foot>
    </div>
</template>

<script>
    import { mapActions, mapGetters } from 'vuex';
    export default {
        mounted() {
            this.SET_BLOG_DATA(); // 调用接口获取数据
        },
        computed: {
            ...mapGetters(['articleList']), // 监听 state
        }
        methods: {
            ...mapActions(['SET_BLOG_DATA', 'SET_NAV_COUNT']),
            countFn(item) {

                // 调用接口存储导航点击次数并跳转，通过派发 action 的形式来发起 state 变化
                this.SET_NAV_COUNT({ type: item.type });
```

```
        }
    }
</script>
```

以上是首页容器组件中的主要代码，其主要做了两件事情：数据的传递和回调的处理，当然还可以包括处理一些该页面中不属于任何一个展示组件的方法，比如校验登录状态。在一个容器组件中可以包含多个展示组件，下面我们来看一下展示组件 Navigation 中的代码：

```html
<template>
    <ul>
        <li
            v-for="(item, index) in nav"
            :key="index"
            @click="goNav(item)"
            v-text="item.name"
        ></li>
    </ul>
</template>

<script>
    export default {
        data() {
            return {
                nav: [{
                    name: '首页',
                    route: 'index',
                    type: 'index'
                }, {
                    name: '文章',
                    route: 'article',
                    type: 'article'
                }, {
                    name: '关于',
                    route: 'about',
                    type: 'about'
                }]
            }
        },
        methods: {
            goNav(item) {
                this.$emit('count', item); // 触发回调
```

```
</script>
```

`Navigation` 导航组件只负责自己内部的数据渲染和回调逻辑，对于存储每个导航的点击量及跳转逻辑来说，作为展示组件这并不是其所关心的，所以我们需要通过触发容器组件回调的方式来实现。再来看一下展示组件 `Article` 的代码：

```html
<template>
    <ul>
        <li
            v-for="(item, index) in list"
            :key="index"
            @click="goPage(item.id)"
            v-text="item.title"
        ></li>
    </ul>
</template>

<script>
    export default {
        props: {

            // 接收容器组件数据
            list: {
                default: [],
                type: Array
            }
        }
    }
</script>
```

展示组件Article中动态的数据通过 `props` 从父组件中获取，其内部只处理文章列表的渲染工作，这样很好的将UI层面和应用层面进行了分离，便于今后该组件的复用。

此外 `Foot` 组件为纯静态组件，其只负责内部数据的渲染，不接收外部的数据和回调方法，这里就不做介绍了。

从以上代码示例中我们不难发现容器组件和展示组件的主要区别和注意点：

作用	描述如何展现（骨架、样式）	描述如何运行（数据获取、状态更新）
是否使用 Vuex	否	是
数据来源	props	监听 Vuex state
数据修改	从 props 调用回调函数	向 Vuex 派发 actions

相比较如果上述的博客首页不做组件的划分，全部逻辑都放在一个组件中，那么必然会导致代码的臃肿和难以维护，而一旦划分了容器组件和展示组件，后期如果哪个页面同样需要展示文章列表，我们只需要传递不同的数据直接复用即可。

组件的层次结构

了解了组件职能的划分后，我们再来看一下组件的层次结构。关于组件的层次，一般页面中不宜嵌套超过 3 层的组件，因为超过 3 层后父子组件的通信就会变得相对困难，不利于项目的开发和维护。3 层结构的容器组件与展示组件的数据传递如下：

可见组件的层次越深数据传递的过程就会变得越复杂，当然这取决于你如何划分容器组件和展示组件，比如我们可以将上述博客首页换一种划分方式：

上图我们页面中存在 3 个容器组件，每个容器组件又可以包含各自的展示组件，这样一定程度上可以减少组件的层次嵌套深度。当然展示组件中也可以包含对应的容器组件来解决数据传输的问题：

这样展示组件 B 下面的容器组件 C 便可以不依赖于容器组件 A 的数据，其可以单独的进行数据获取和状态更新。

而对于那些你不知道应该划分为容器组件和展示组件的组件，比如一些耦合度较高的组件，那么你可以暂时归类到其他组件中，混用容器和展示，随着日后功能的逐渐清晰，我们再将其进行划分。

结语

本文主要介绍了容器组件和展示组件的概念和层次划分，在编码上，容器组件和展示组件各司其职，它们将容器和展示更好的分离，提高了组件的重用度，降低了功能上的耦合度，为高效、高质量的代码开发奠定了基础。

思考 & 作业

- 如果需要你对掘金首页进行组件的划分，你会如何划分其结构和层次？
- 在子组件的 `props` 中，如何动态的设置默认值?

留言

评论将在后台进行审核，审核通过后对所有人可见

chenlinfei .NET攻城狮 @ 深圳某互联网金融科技...

学习vue不久，第一次了解到组件有容器组件和展示组件之分，赞.jpg

▲ 0　　评论　16天前

雕刻时光本尊

这节写的真心不错，将划分组件的思想和主要方法都写出来了。之前没思考过，React中的展示组件和容器组件的思想和方法，其实是组件化开发的通用的思想和方法。跟着作者学习，能打通好多关节。

▲ 0　　评论　2月前

给力小青年 前端小白菜

----父组件

<navigation @count="countFn"></navigation>

----子组件

goNav(item) {

this.$emit('countFn', item); // 触发回调

}

这个父组件监听子组件的事件。。。是不是写反了?

▲ 0　　收起评论　3月前

劳卜 前端工程师 @ TC

确实，已修正

3月前

雪飞花1991 前端开发工程师 @ 光环云

这个能不能再深入一下。 现在我有个页面，页面分五块，五个form表单，我抽出为五个组件。 这五个组件我又分为 所谓的容器组件和展示组件，容器组件负责数据业务处理，展示组件负责样式及内部逻辑处理。但是这样就有三层嵌套，导致最外层父组件调用其孙组件时，相当困难。 期间我使用mixin进行公共方法抽离。 感觉有些混乱

▲ 0　收起评论　3月前

劳卜 前端工程师 @ TC

使用 Vuex 来统一处理数据会更好一点

3月前

评论审核通过后显示

评论