Atitit 组件化规范组件化的艺术 attilax著

目录

[1. 重要概念 1](#_Toc26627)

[2. React认为一个组件应该具有如下特征： 1](#_Toc10432)

[2.1. 可组合（Composeable）： 1](#_Toc6759)

[2.2. 可重用（Reusable）： 2](#_Toc13325)

[2.3. 可维护（Maintainable） 2](#_Toc15548)

[3. Handlebars或Mustache等模板引擎 2](#_Toc32119)

[4. 局部组件化”，“全组件化”。 2](#_Toc22500)

[5. 组件化的本质目的并不一定是要为了可复用，而是提升可维护性。 不具有复用性的组件” 3](#_Toc11255)

[6. 组件化实现方式 3](#_Toc6686)

[7. Html组件化的标签嵌套层次3到5层 4](#_Toc4919)

# 重要概念

Web Components

* 通过shadow DOM封装组件的内部结构
* 通过Custom Element对外提供组件的标签
* 通过Template Element定义组件的HTML模板
* 通过HTML imports控制组件的依赖加载

# React认为一个组件应该具有如下特征：

## 可组合（Composeable）：

一个组件易于和其它组件一起使用，或者嵌套在另一个组件内部。如果一个组件内部创建了另一个组件，那么说父组件拥有（own）它创建的子组件，通过这个特性，一个复杂的UI可以拆分成多个简单的UI组件；

## 可重用（Reusable）：

每个组件都是具有独立功能的，它可以被使用在多个UI场景；

## ****可维护（Maintainable）****

1. ：每个小的组件仅仅包含自身的逻辑，更容易被理解和维护；

# Handlebars或Mustache等模板引擎

Handlebars或Mustache等模板引擎来处理视图层。但React相信视图和控制器应该相互依存在一起而不是使用第三方模板引擎

# 局部组件化”，“全组件化”。

我们来看看如何把一个业务界面切割成组件。

有这么一个简单场景：一个雇员列表界面包括两个部分，雇员表格和用于填写雇员信息的表单。在这个场景下，存在哪些组件？

对于这个问题，主要存在两种倾向，一种是仅仅把“控件”和比较有通用性的东西封装成组件，另外一种是整个应用都组件化。

对前一种方式来说，这里面只存在数据表格这么一个组件。  
对后一种方式来说，这里面有可能存在：数据表格，雇员表单，甚至还包括雇员列表界面这么一个更大的组件。

这两种方式，就是我们之前所说的“局部组件化”，“全组件化”。

# 组件化的本质目的并不一定是要为了可复用，而是提升可维护性。 不具有复用性的组件”

大量的业务界面，这块东西很显然复用价值很低，基本不存在复用性，但仍然有很多方案中把它们“组件化”了，使得它们成为了“不具有复用性的组件”。为什么会出现这种情况呢？

组件化的本质目的并不一定是要为了可复用，而是提升可维护性。这一点正如面向对象语言，Java要比C++纯粹，因为它不允许例外情况的出现，连main函数都必须写到某个类里，所以Java是纯面向对象语言，而C++不是。

另外有一些框架/库偏爱用函数逻辑来生成界面，早期的ExtJS，现在的React（它内部还是可能使用模板，而且对外提供的是组件创建接口的进一步封装——jsx）等，这种实现技术的优势是不同平台上编程体验一致，甚至可以给每种平台封装相同的组件，调用方轻松写一份代码，在Web和不同Native平台上可用。但这种方式也有比较麻烦的地方，那就是界面调整比较繁琐。

# 组件化实现方式

#### **Iframe 容器化**

还记得iframe们吗？我们还在使用它们，是因为他们能确保组件和控件的JavaScript和CSS不会影响页面。 [Shadow DOM](http://www.html5rocks.com/en/tutorials/webcomponents/shadowdom/) 也能提供这样的保护，并且没有iframe带来的负担。正式的说法是：

Shadow DOM的设计是在shadow根下隐藏DOM子树从而提供封装机制。它提供了建立和保障DOM树之间的功能界限，以及给这些树提供交互的功能，从而在DOM树上提供了更好的功能封装。

#### **HTML导入**

我们长时间以前就可以导入JavaScript和CSS了。 [HTML导入功能](http://www.html5rocks.com/en/tutorials/webcomponents/imports/)提供了从其他HTML文档中导入和重用HTML文档的能力。这种简单性同时意味着可以很方便地用一些组件构建另一些组件。

最后，这样的格式很理想，适合可重用组件，并且可以用你最喜欢的包管理解决方案发布（例如： [bower](http://bower.io/)、 [npm](https://www.npmjs.org/) 或者 [Component](https://github.com/component/guide)）。

# Html组件化的标签嵌套层次3到5层

****把什么视为组件，组件的粒度是怎样的。****

在组件化的应用中，组件树的层级不宜过深，从根节点算起，应当尽可能控制在3到5层内，如果层级太多的话，会造成组件通讯和数据传递的负担。

Ref

atitit.组件化原理与设计 原则 v2 q48

atitit.atiHtmlUi web组件化方案与规范v1

Atitti.组件化的设计与原理 大纲

参考

Atitit..组件化事件化的编程模型--(2)---------Web datagridview 服务器端控件的实现原理and总结

Atitit.jsp web js 组件化设计

Atitit.ui开发中的 重量级组件

Atitit.组件化的趋势，与设计原则

**组件和服务**