CocosCreator组件化编程的探索

使用CocosCreator已经一年了，在此期间一直在摸索，如何才是组件化编程的最优实践。Shawn属于半野生的路子，水平不高，但不时会陷入一些问题瞎琢磨。我根据自己的经验，总结了一套组件化编程模型：**法宝**与**结界**模型。

但在介绍**法宝**、**结界**组件模型之前，先回顾一下控件和组件的概念。

一、控件与组件

在Cocos2d-x/lua/js的年代，UI元素都以控件类的形式存在。只有cc.Node的子类才能在界面上显示，例如：cc.Image、cc.Button、cc.Text。我们编写的界面代码也属于cc.Node的子类或系统控件的子类。

在Creator中我们自己编写的cc.Component的子类脚本，能称之为控件吗？



1.我所理解的控件

我觉得要能称的上控件，必须是能够被**界面编辑器**和**代码**所控制，并能相对独立完成一项或多项任务的程序模块。而且控件具有一定范围的通用性，可以独立运行，可以被独立测试。

我们要自定控件，一类是**cc.Componet子类脚本** + **预制体**的结合；第二类是**纯cc.Componet的子类脚本**（不含预制体），也可以是系统组件的派生类。不含预制体的脚本其实是引擎自动帮我们生成的，当拖入一个组件脚本到场景编辑器，Creator会自动生成一个节点，并将脚本挂载到这个节点上。

这两类控件有什么不同呢？

脚本+预制体：控制的是预制体中的节点和子节点，以及节点上的控件。

纯脚本: 只能控制当前节点，也可以控制当前节点上的其它组件。

可以看出，这两类组件代码在他们控制的范围上是不同的。

2. 组件的悲剧

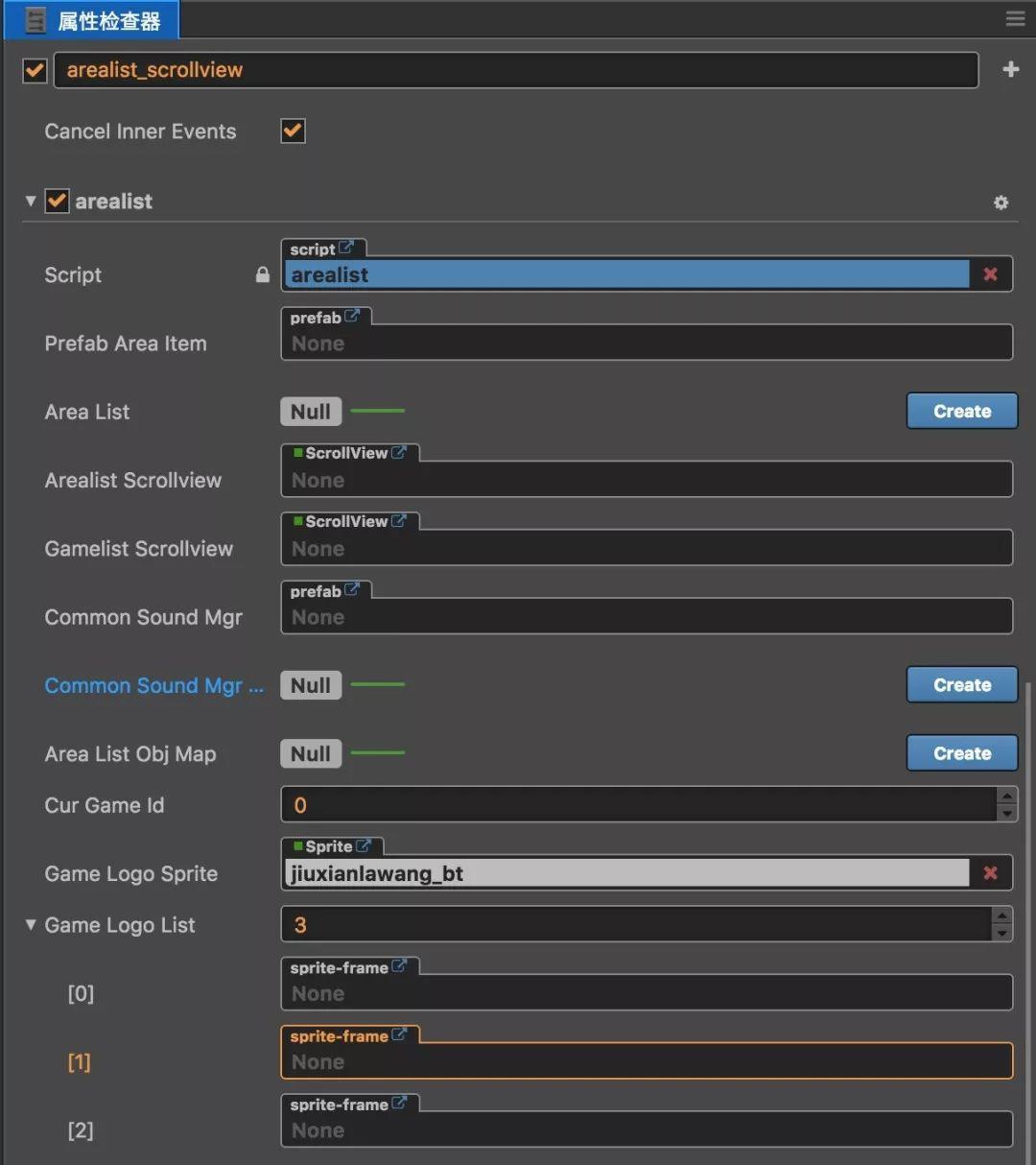
cc.Componet的子类都是组件，但他们要想要上升成为控件却很难。因为大多组件代码，都无法像系统控件那样独立完成一项目任务，其原因之一是滥用组件的**properties可视化编辑功能**，将本职范围内的节点做为成员变量，目的仅仅是为了方便访问。

[图片上传失败…(image-346e88-1518414298577)]

从Creator范例工程中的TestList首场景为例，Menu.js组件脚本挂载到Menu节点上，最后一个Menu.testList属性设置是非当前节点的子节点，控制权延生到了外面去了。前面几个属性（Text, Readme…）都是通过编辑器拖拽将Menu节点下的几个子节点配置到了组件脚本上，他们对于Menu.js应该属于私有成员变量，也变成了公开的了。

通过简单拖拽配置成员属性确实让程序开发变的简单，但如果滥用会有一个严重的问题：控件属性由原来的点状（控制自身节点）或线性（控制子节点或成员节点）关系，变成了网状关系（控制自身以外的节点）。

这将导致组件脚本难以独立完成任务和测试，必须通过编辑器正确配置才能工作，就像在一个模块代码中访问了全局变量一样。要让Menu.js成为控件的办法，最好是将TestList节点放到Menu节点内部。把Menu节点拖到资源管理器中成为一个prefab。



不知道如何下手，设置这些属性

不知道大家有没有遇到过，在属性检查器上密密麻麻的属性配置，不知道该如何下手？更让人头痛的是，不小心代码冲突，导致组件属性配置丢了，再看代码，脚本中的属性变量与节点名字又对应不上，就连编写这个模块代码的人都搞不懂是怎么会事！

网状关系的程序组织结构，会导至模块之间相互依赖，可重用性极低。如何规范组件的编写方式，确保模块的内聚性值得我们多多思考。

二、法宝与结界

下面来聊聊我总结的**法宝**与**结界**模型，假想一个完整的世界，为了维护这个世界的有序运行，设置了一个结界。结界中有无数的法宝参与到世界的运行之中，贡献出力量。

1.法宝型组件

法宝型组件：以装饰宿主节点为己任，从不控制其它节点。

法宝型的特点是通用性强，可挂载任意节点运行，Creator内置的组件绝大多数属于这类。例如有Sprite、Label、Button、Widget等，可以看出纯脚本的组件就属于法宝型。

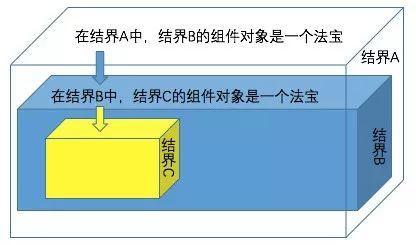
2.结界型组件

结界型组件：管理和控制其它节点及节点上的组件，通常会根据上层业务要求，调用其它节点的属性方法完成任务。

结界型的特点是业务逻辑性强，通用性差，通常是对法宝型组件的指挥和管理。组件+prefab就属于这类，由于结界型组件大多是定制的，它并且不能随便挂载到别的节点上（更多的是只能挂载到唯一的节点上）。

3. 结界的秘密

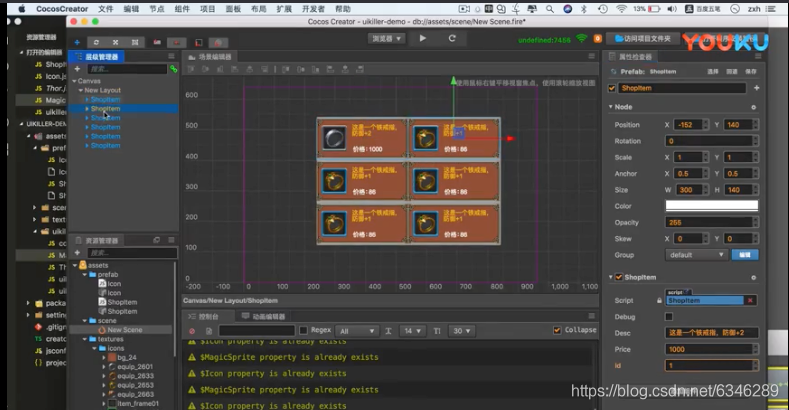
话说天有九重，九只是个虚数，其实是很多的意思。一个结界型组件，对于它的上层结界来说，他又是一个法宝型组件，这就形成了模块化。



比如有一个名为A的prefab，将组件脚本A.js挂载到prefab的根结点。当另一个场景或预制体中实例化这个A.prefab时，A.js就上升为一个控件，他管理了A.prefab下的所有节点，但对于当前场景来说，它又体现为一个法宝型组件，而成为了一个控件。

对内是结界，从根节点开始自下而下管理所有子节点；对外是法宝，从根节点获取法宝暴露的属性方法。这样以内部线性、总体树状的程序结构，是不是要比网状的结构更好些呢？

三、小结

[uikiller](https://github.com/ShawnZhang2015/uikiller)库是我在组件化编程上的一点成果，可以方便管理prefab下的任意节点和组件，以及节点上的触摸事件。奉上一段uikill的使用视频 ，结束这篇分享。  
  
[雷神归来-Thor组件实战演示](http://v.youku.com/v_show/id_XMzMxNzUxMzYzMg==.html?spm=a2hzp.8244740.0.0)