什么是Webcomponent?

Webcomponent是一套允许你创建出可重用element的技术。创建出来的element,叫”custom element”,它内部的功能和样式是独立于页面中的其它部分的，你可以在其中使用html原生的dom元素构建出一套完整的样式及功能。在以往没有webcomponent技术的时候，要创建一个可重用的ui组件,且保证在多次使用它的时候不会发生错乱是一件困难的事。

Webcomponent技术说白了，就是用HTML原生的DOM元素组合封装出来一个新的元素。

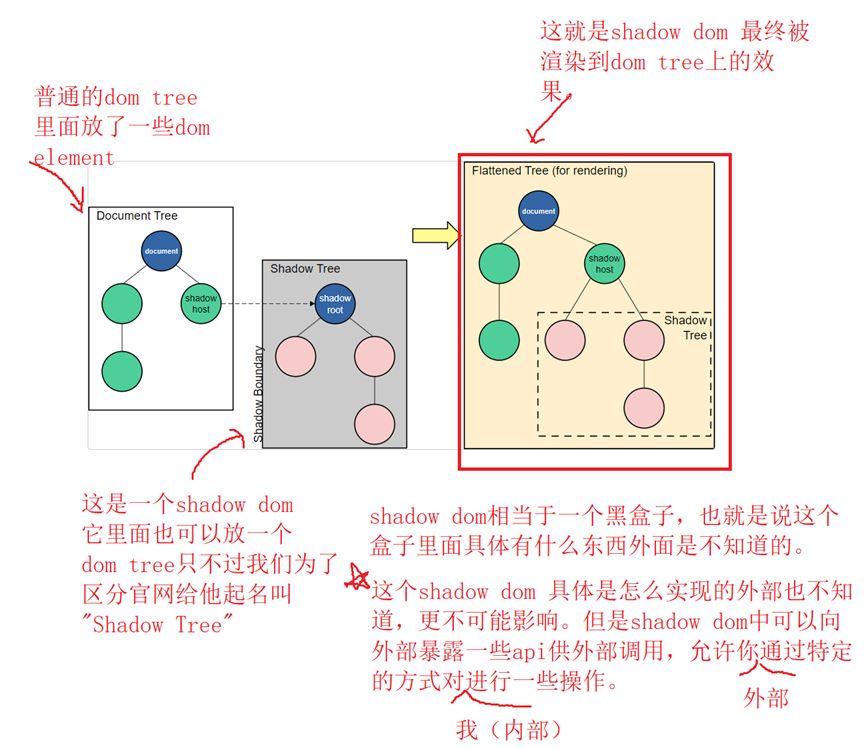
为什么要学习Webcomponent？

在前端开发中，你可能会用到第三方的ui组件库，这些组件库都是基于webcomponent实现的。有的时候呢，你可能会需要基于人家的组件做一些自定义样式或功能的操作。或者要做一些适配工作。那，如果你不了解webcomponent的话，你可能不太好做上述那些工作。另外，有时候，第三方组件库提供的ui组件，不能满足你的需求，且你需要自定义一个组件，以达到重复使用的目的的时候，你就需要用到webcomponent了。

Webcomponent中的几个比较重要的概念。

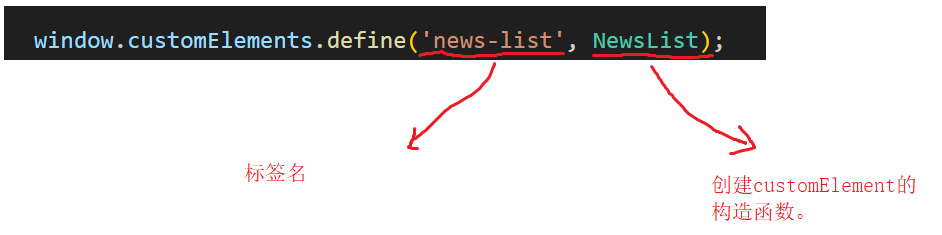
1. Shadow DOM Tree

前面我们提到过，创建一个可复用的ui组件，最终要的是如何保证全局css样式和Js对组件内部的功能不产生影响。而这个组件，最终也是要放到页面上渲染的。只不过它内部的结构和逻辑不受外部的影响，为此，Webcomponent提供了一个新的概念，“Shadow DOM Tree”。它好比一个黑盒子，在这个黑盒子中，有一个像“document Tree”结构的Tree,叫”shadow Tree”,里面放了很多html的普通dom元素。Webcomponent给我们提供了操作”shadow Tree”里DOM元素的api，让我们可以在组件的内部操作shadow DOM里的元素，以实现特定的功能。同时我们也可以使用css在Shadow Tree内部实现对Shadow DOM样式的美化。



找到一个Shadow DOM Tree的宿主元素

我们可以直接使用window全局对象的customElements.define()方法定义一个shadow host,也就是所谓的shadow dom tree的宿主元素。在html中，它其实就是一个标签，只不过这个标签名是你自定的（这个标签名作为shaodowDOM的宿主，也就是shadowhost，是需要你在html中提前定义的）。

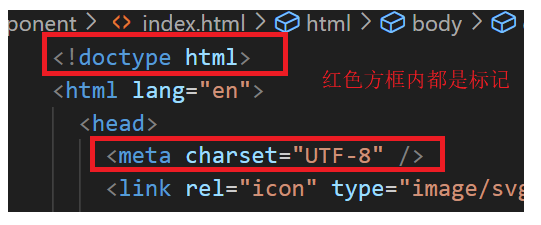


有关这个方法的使用细节可以参看mdn：<https://developer.mozilla.org/en-US/docs/Web/API/CustomElementRegistry/define#description>

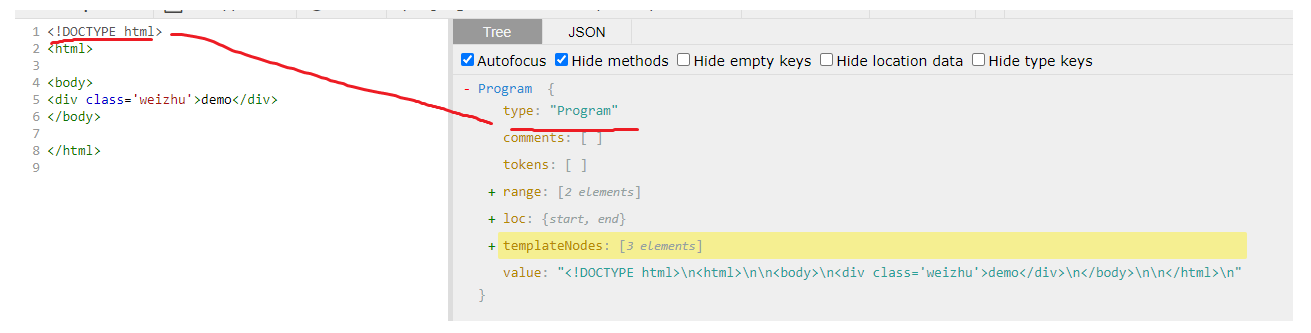
Define方法的第二个参数是一个构造函数，用于构建customElement。你可以使用es6给我们提供的语法糖来完成构建customElement。但是，请注意：无论你使用那种方式，你都需要将构造函数的原型指向HTMLElement构造函数，使用class则可以直接使用extend关键字继承，比较方便。

有关customElements.define()，第二个参数的介绍。

如果你对为什么要继承HTMLElement构造函数有困惑，那可能是你对HTML的Node节点和HTMLDOM元素有些傻傻分不清，我在这里简短的解释一下：Node节点是几乎页面上的所有元素，包括whitespace,还有一些不参与页面布局的标签。因为，\*.html是一个文件，这个文件中有很多文本内容，解析器在解析它的时候呢，需要知道我应该如何解析这个文档，所以你会在Html文件中看到一些貌似与web开发无关的标签，其实它就是一个告诉解析接下来的文档的标志，就像下图：



这些所有的元素，都可以说是Node节点，这些节点会被抽象成一个对象。它可以有自己的子集和父级。

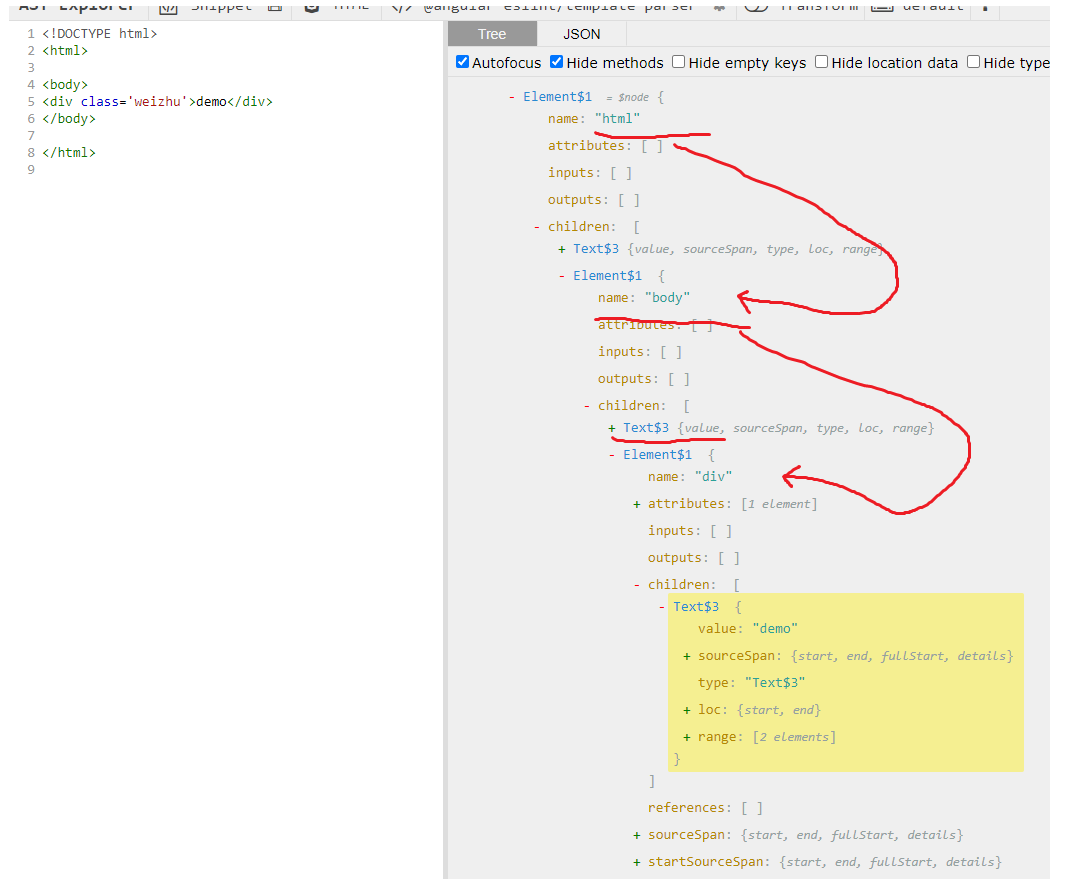


解析器就是通过依序读取并解析这些标签生成一个ast语法树，解析html文件中的内容并生成ast语法树的大致步骤是：



所谓ast语法树，就是将html中的标签抽象成一个对象描述它，只不过它是树状结构的。

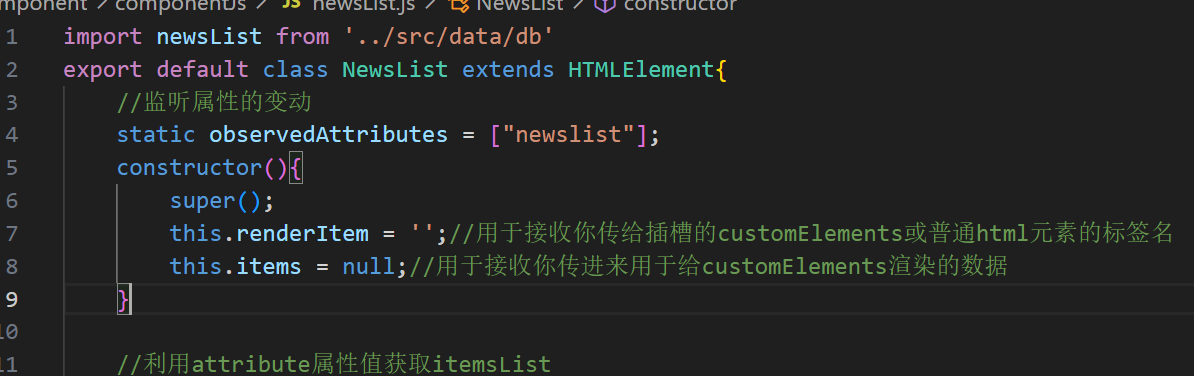
如下图所示：



所以DOM元素对象，你可以说它是Node节点的子集，是由Node节点对象进一步抽象封装而形成的对象，所以它可以用Node节点的api，也有自己的api。基本上参与页面布局的标签都是DOM对象，它们就是用HTMLElement构造函数生成的。所以我们在创建custom Element的时候，也要继承HTMLElement构造函数。

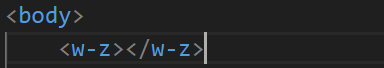
创建生成customElement的类。

这个类的构建其实非常简单，但是其中有一些细节不能在这个小节中一次性讲完，你只需要知道，它是这样构建的就行。



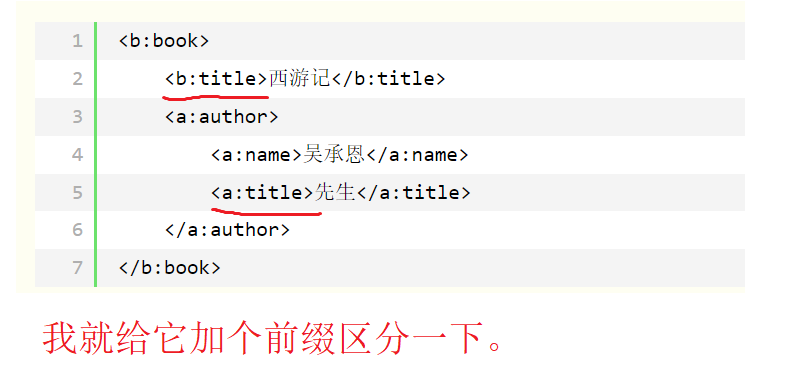
使用上面提到的知识创建一个简单的customElement。

第一步：我们需要在html页面中顶一个shadowhost的标签。

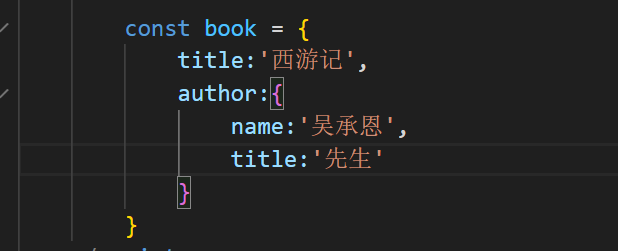


有可能你会对这个产生一定的疑惑，这个需要你对xml文件有有一点点了解，其实我们所谓的html文件，是基于xml文件衍生而来的（它们之间的标签很像，但是它们的解析器不同）。这里我简短解释一下：

在xml中，标签呢，是可以你任意自定的。Xml文件最早呢，是用来传输数据用的：而且是传输有一定格式的数据，例如下图，



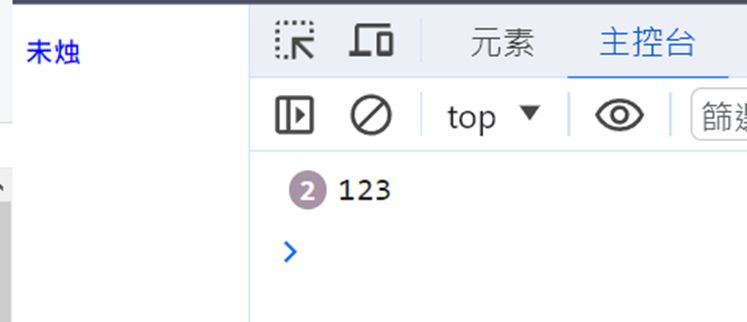
上图的xml文件传输过来的数据最终就会被映射成一个对象（或者说你可以通过对应的xml文件解析器将它其中内容解析出来，它描述了一个类似于Js中对象的数据结构。）如下图所示：



但是在html中，DOM元素标签，都是预定义的，所以我们可以直接用。DOM标签上的属性和标签名等信息都会被抽象成一个对象。后续我们所做的所谓的DOM操作其实就是在操作这个对象。包括我们通过css设置的样式，最终都是被attach到DOM对象上的。我举个例子你就能很快理解：

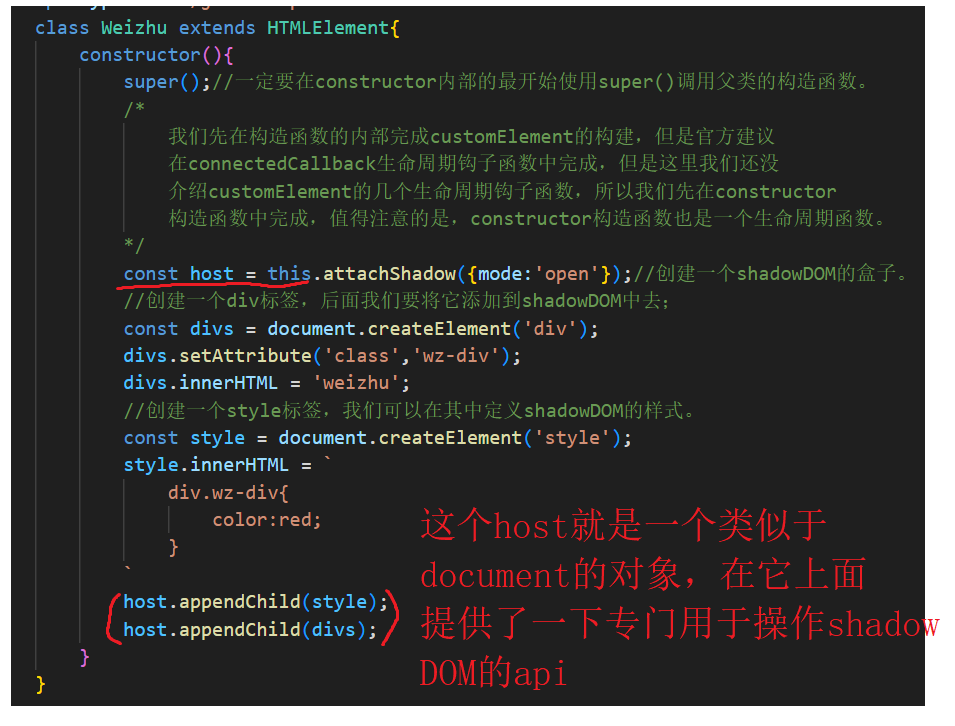


页面的效果：



所以我们定义的<w-z></w-z>shadowhost标签呢，html是不认识它的，但是它确实是一个标签，也是一个节点。我们第后面就是要让html认识它，知道它是一个我们自定义的标签，然后在这个标签中我们给它attach了一个shadowDOM tree，你需要给我渲染出来。

第二步：开启一个shadowDOM的黑盒子，在其中添加dom元素和样式。



最后一步：将我们构造的shadowDOM tree附到我们创建的自定义标签上。



最后我们来查看一下页面的效果：



到这里webComponent的基础知识我们就已经介绍完了，后面我会介绍一下它的一些高阶用法，最后我们尝试自己封装一个ui组件。