Angular组件间的通讯

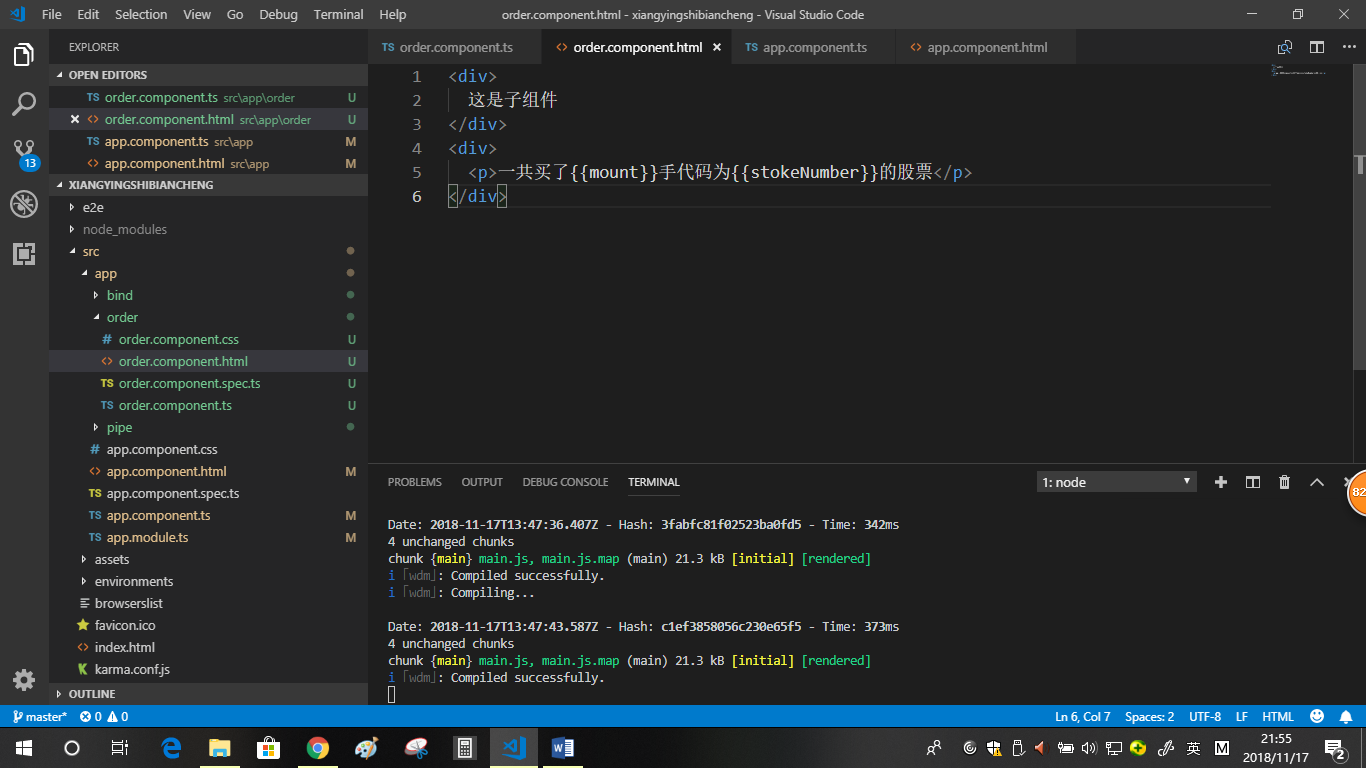
1. 输入属性：（仅仅用于父组件--->子组件之间的数据单向传递）

(1).子组件ts文件：

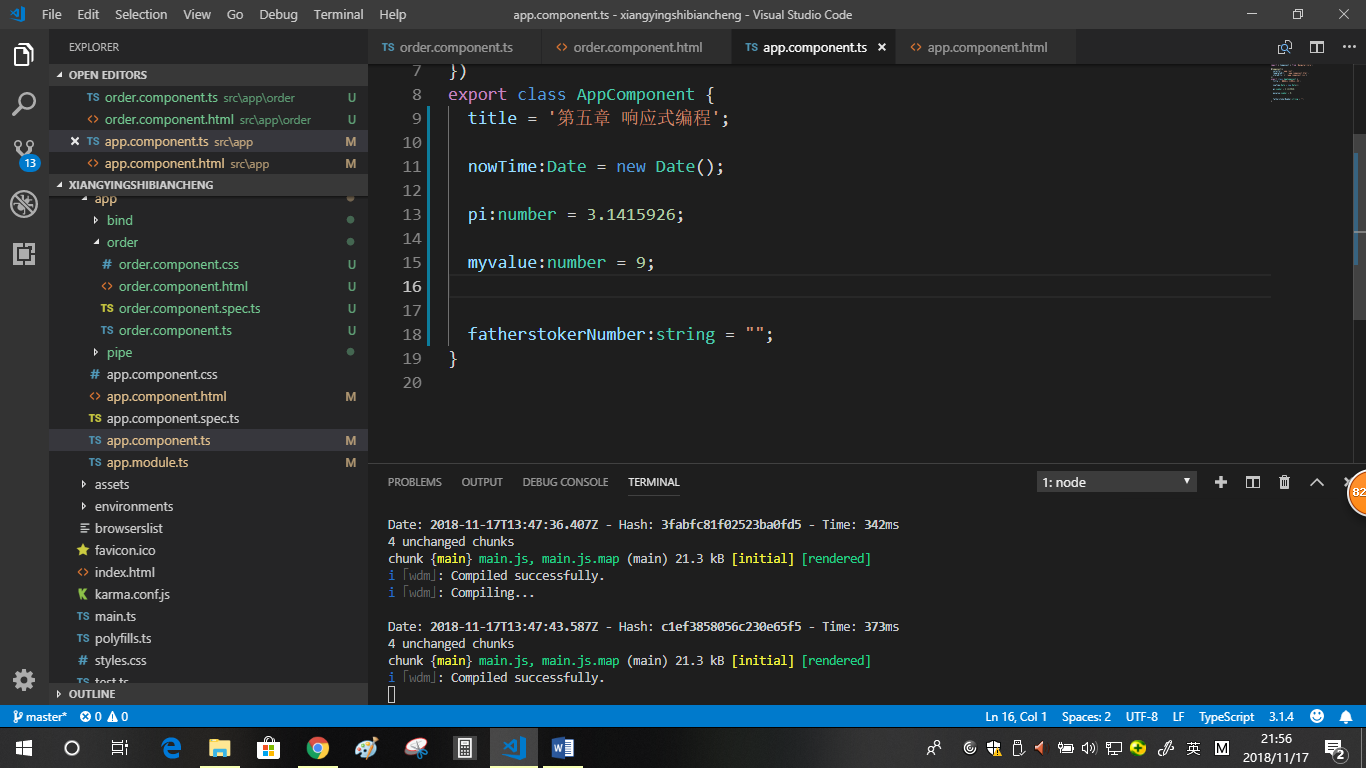
使用@Input()装饰器



（2）.子组件html文件：

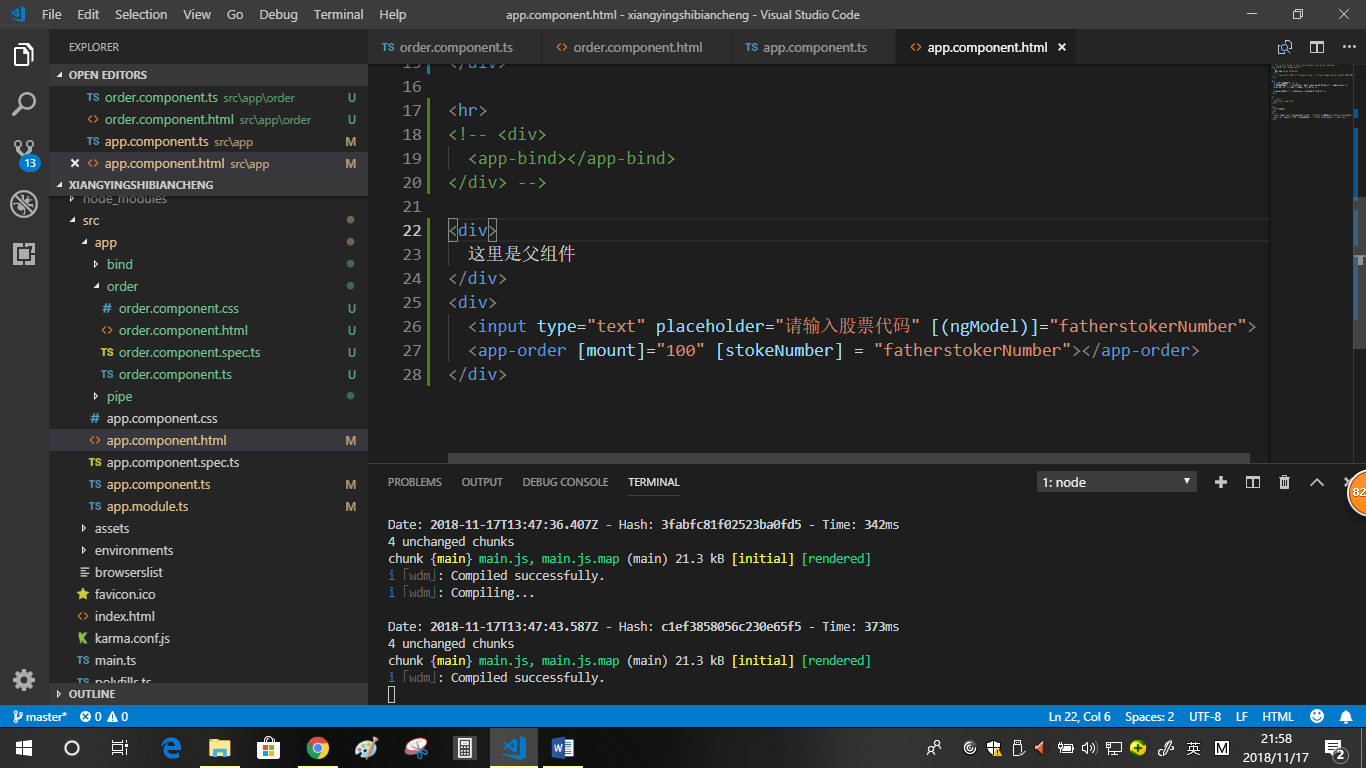


（3）.父组件ts文件：



只是声明了一个用于接收参数的变量

（4）.父组件html文件：



数据双向绑定：需要引入FormsModule

通过给子组件属性绑定传递值过去

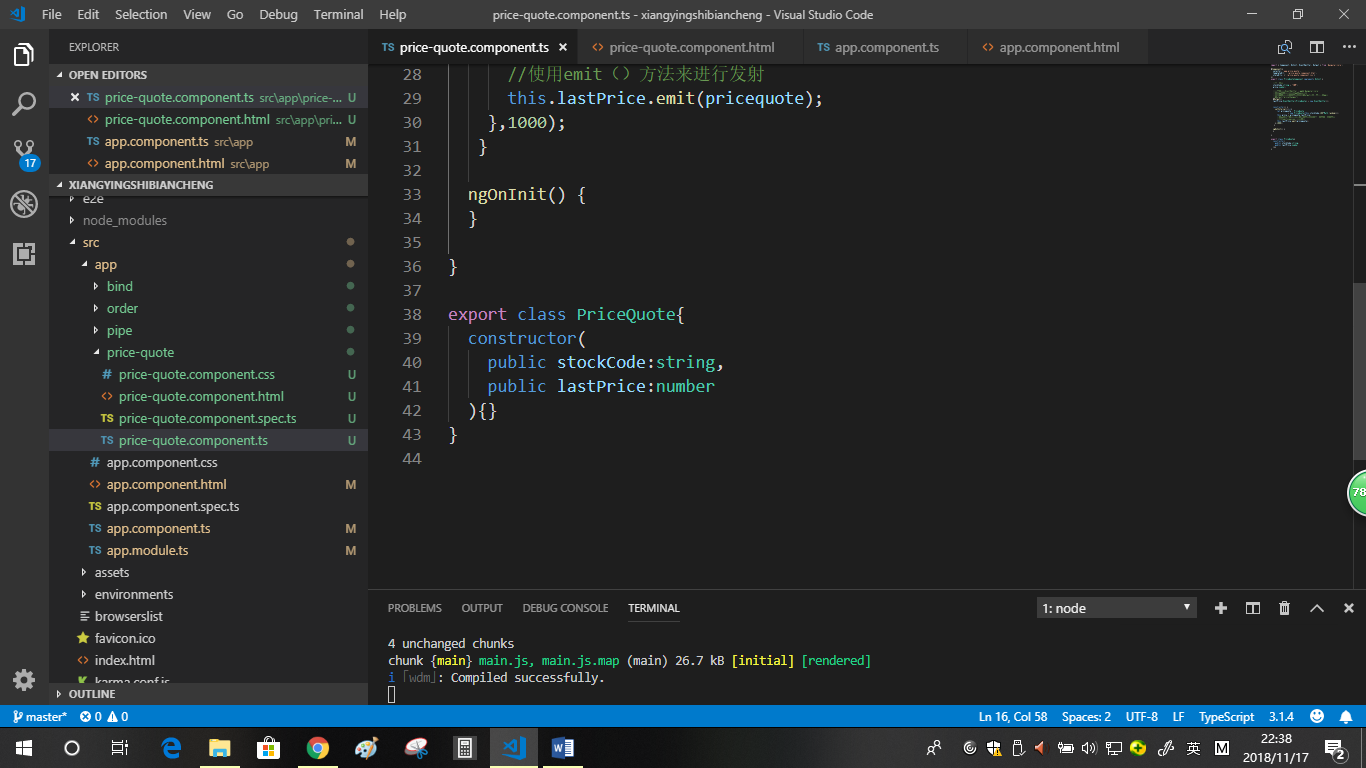
（5）效果图：



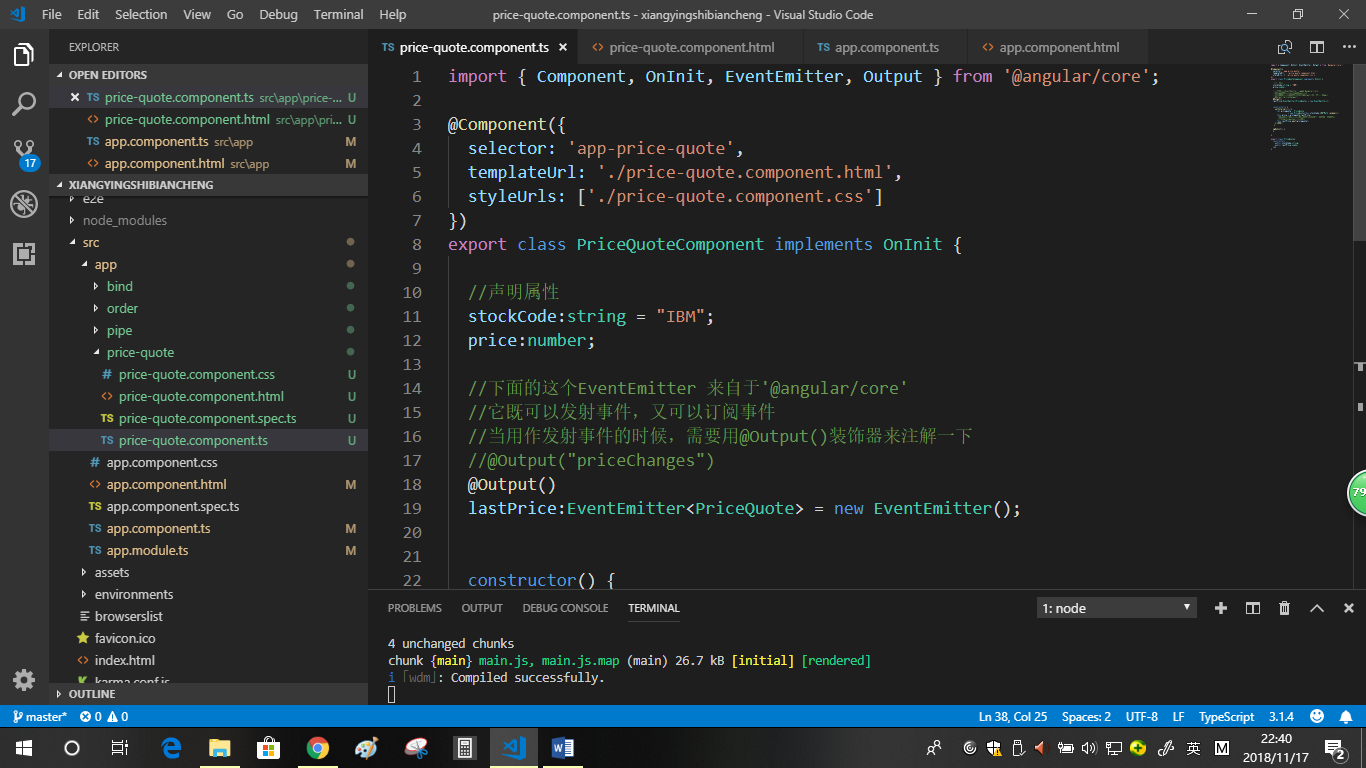
1. 输出属性：从子组件传递数据到父组件

模拟子组件数据的变化传递给父组件的例子：

（1）.子组件的ts文件：

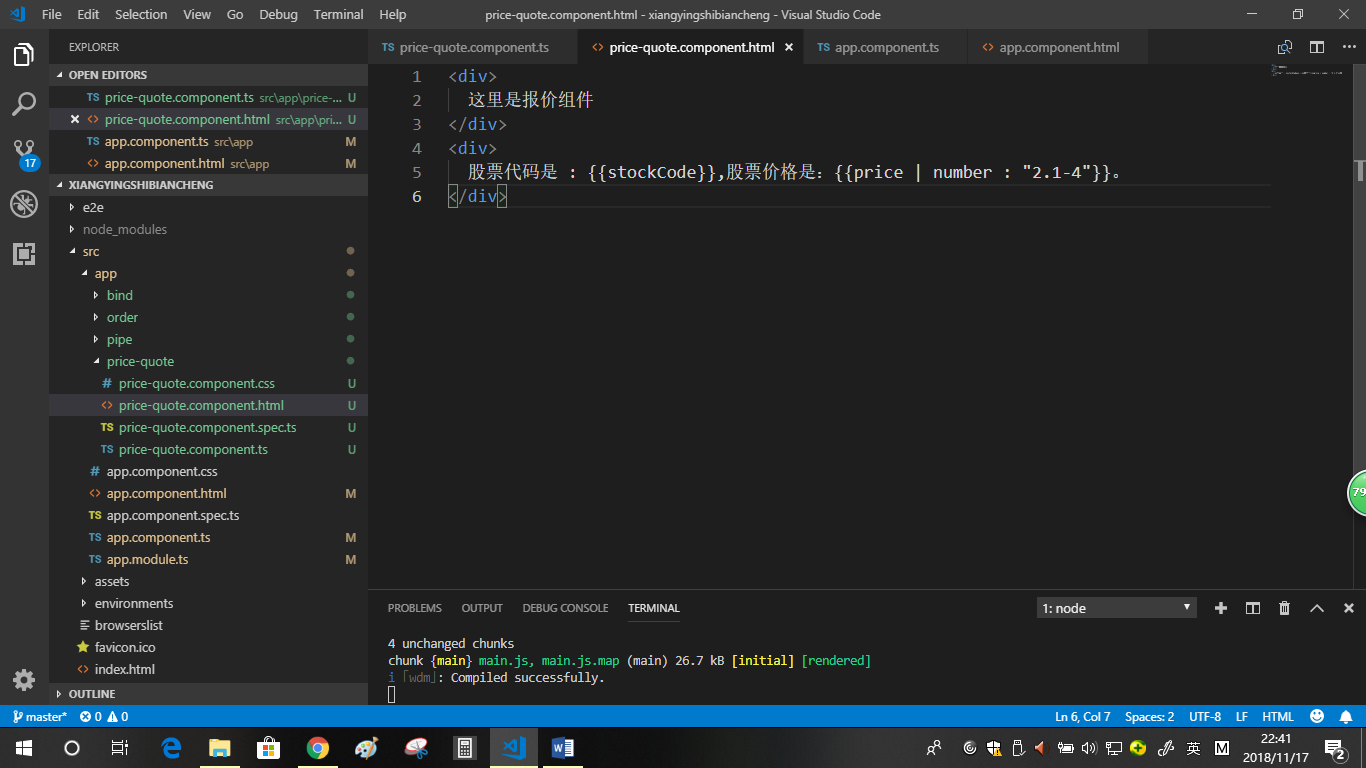


声明一个保存股票代码和股票价格的类



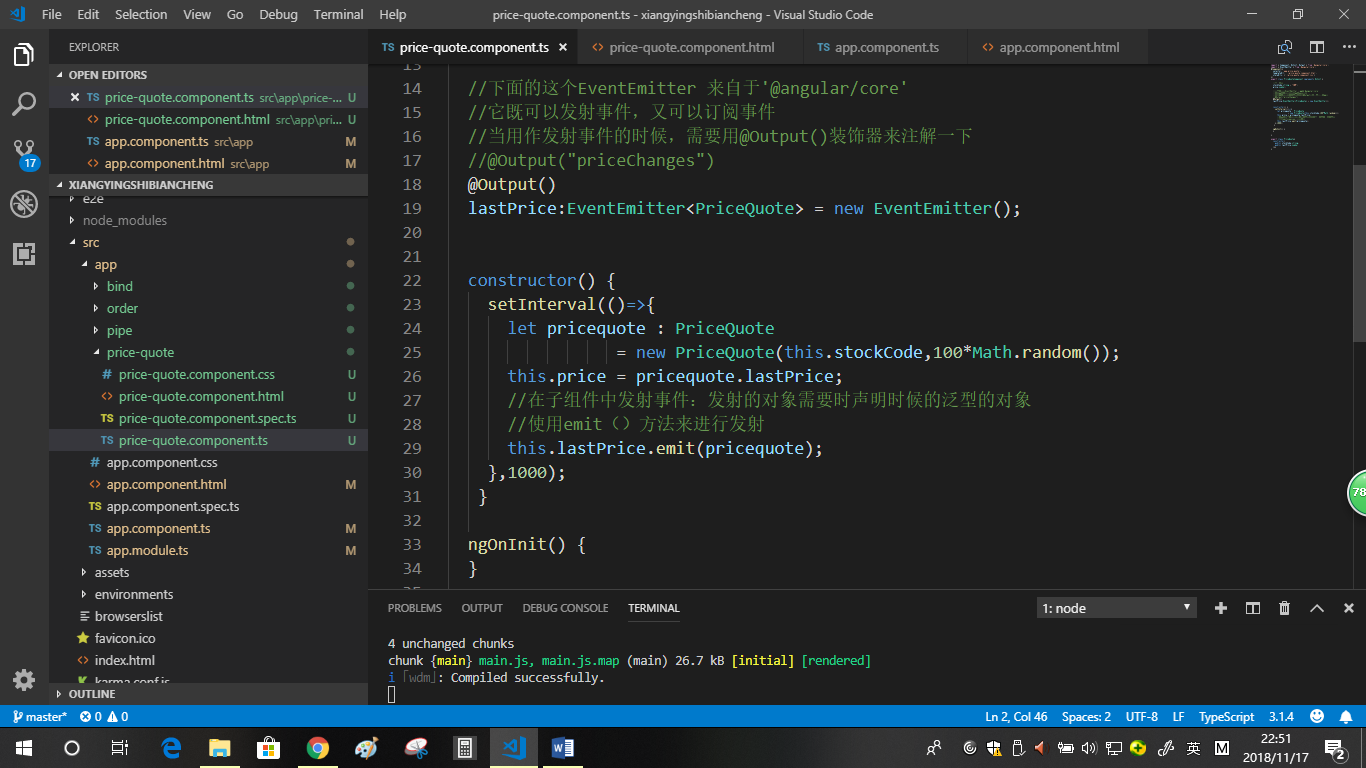
声明属性

（2）子组件中的html文件

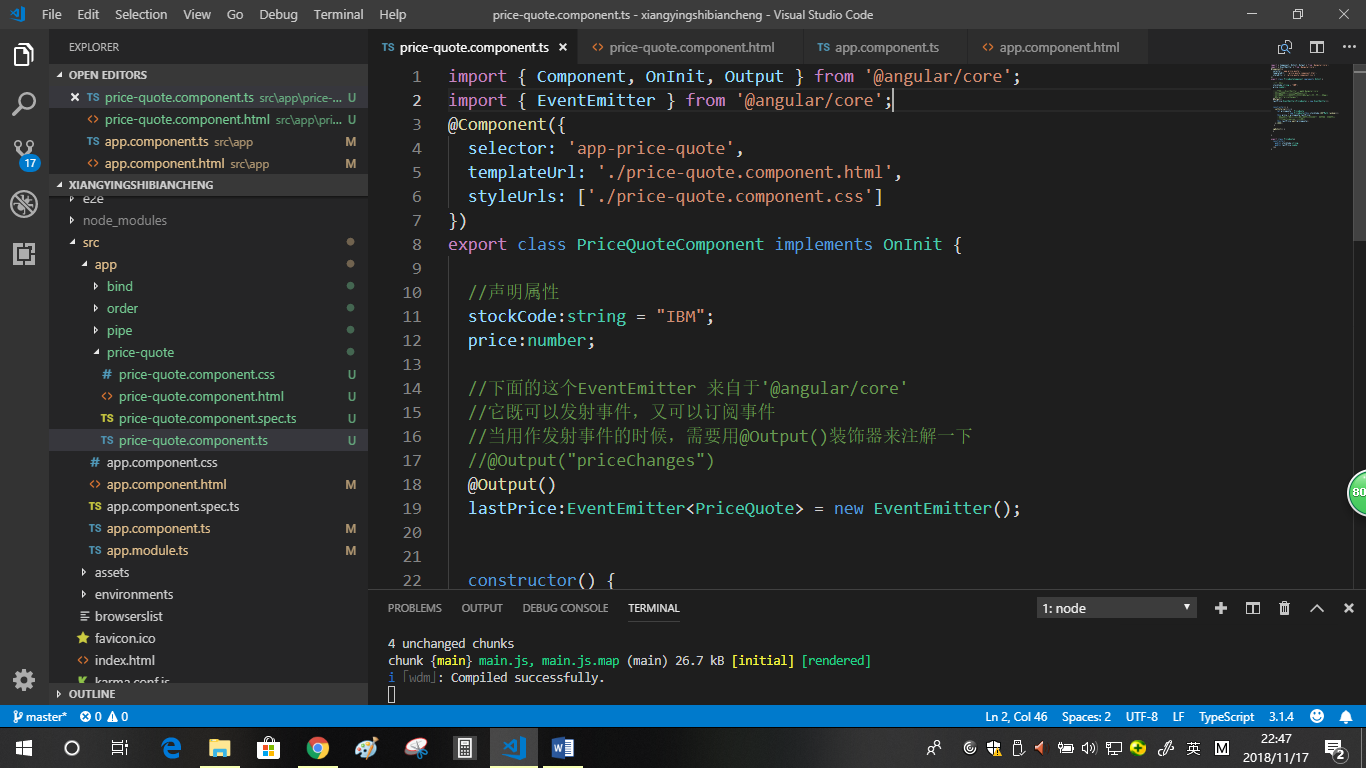


Html文件中的内容

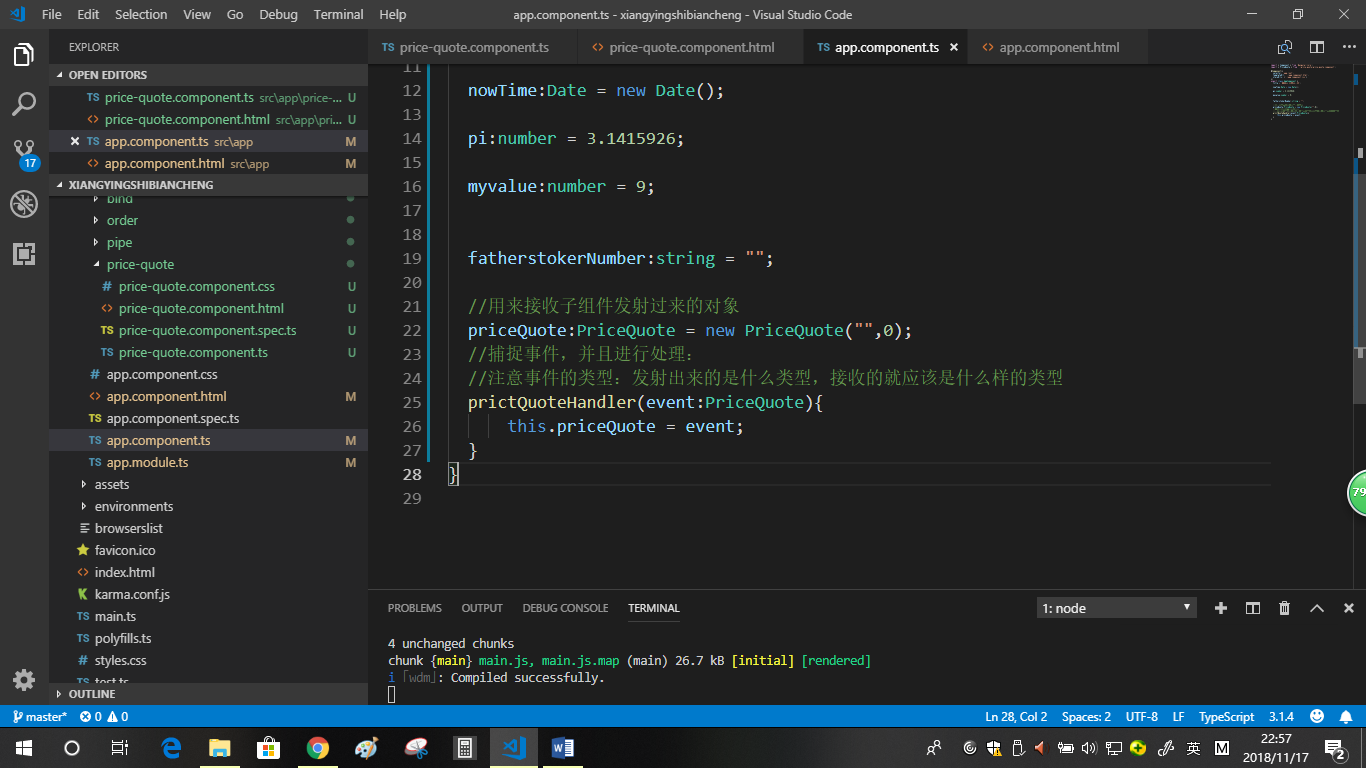
以上都是最基本的操作，下面才是重点

（3）子组件中创建输出属性，并且发射事件：

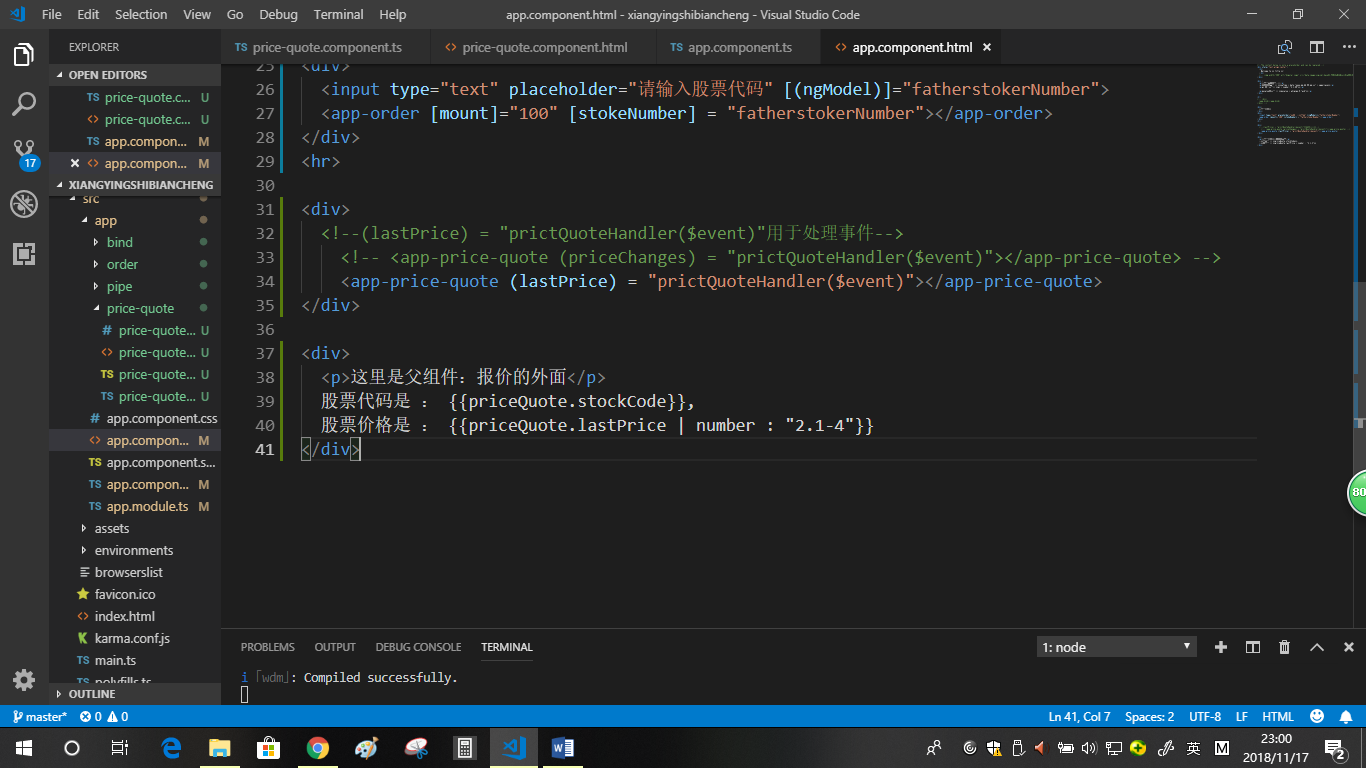
[1.@Output()](mailto:1.@Output())无参数时，事件默认为下面的名字；有参数时，事件为参数的名字，用于父组件捕获事件时用



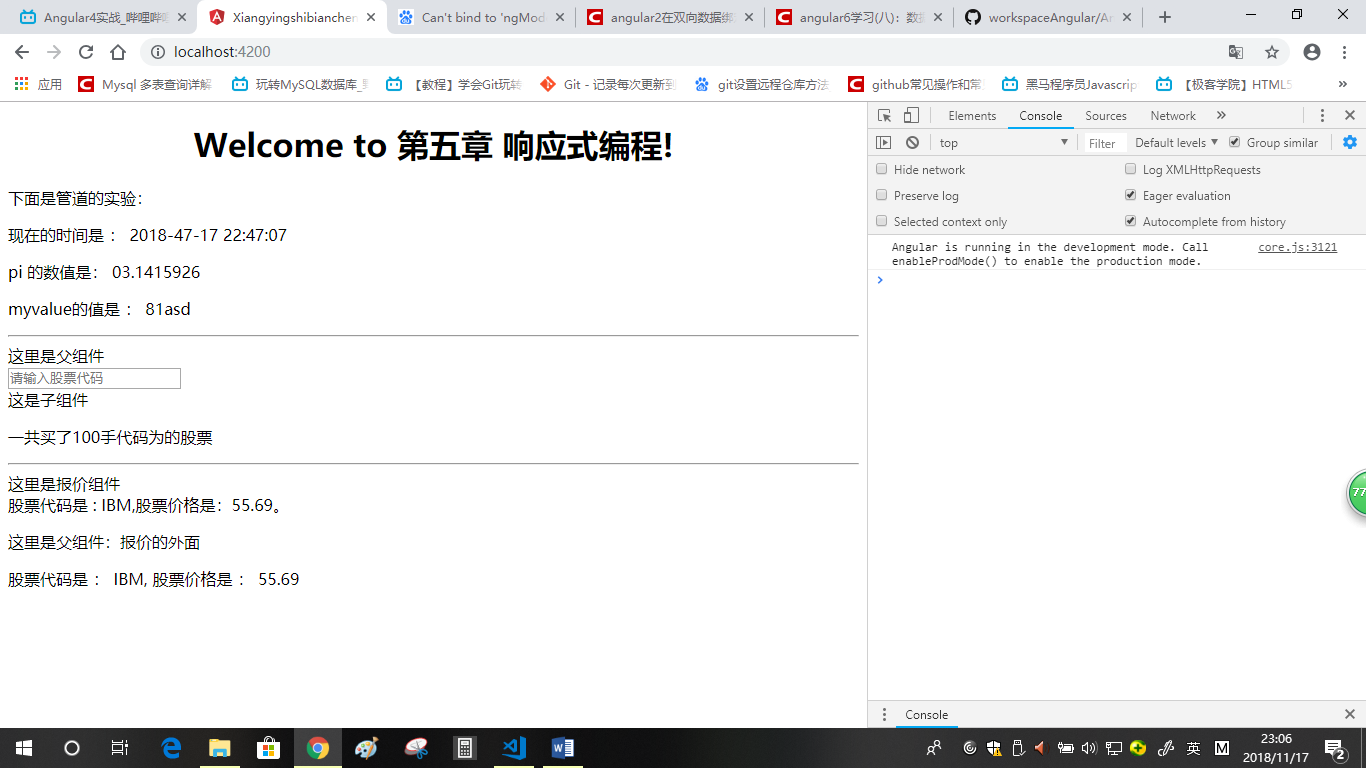
事件在发射事件：emit方法发射事件，事件的参数类型为定义手的泛型的类型

（4）父组件的ts文件：

这里的参数的类型，需要时子组件中发射事件参数的类型：就是emit()方法中的数据类型=》发射什么类型，接收什么类型

（5）父组件的html文件：

注意事件的名称，与子组件中@Output的参数有关系的

（6）这就完了！结果图：

1. 中间人模式

中间人模式就是具有同一个父组件的兄弟之间的数据传递，结合了输入属性和输出属性的应用。

父组件

子组件2

子组件1

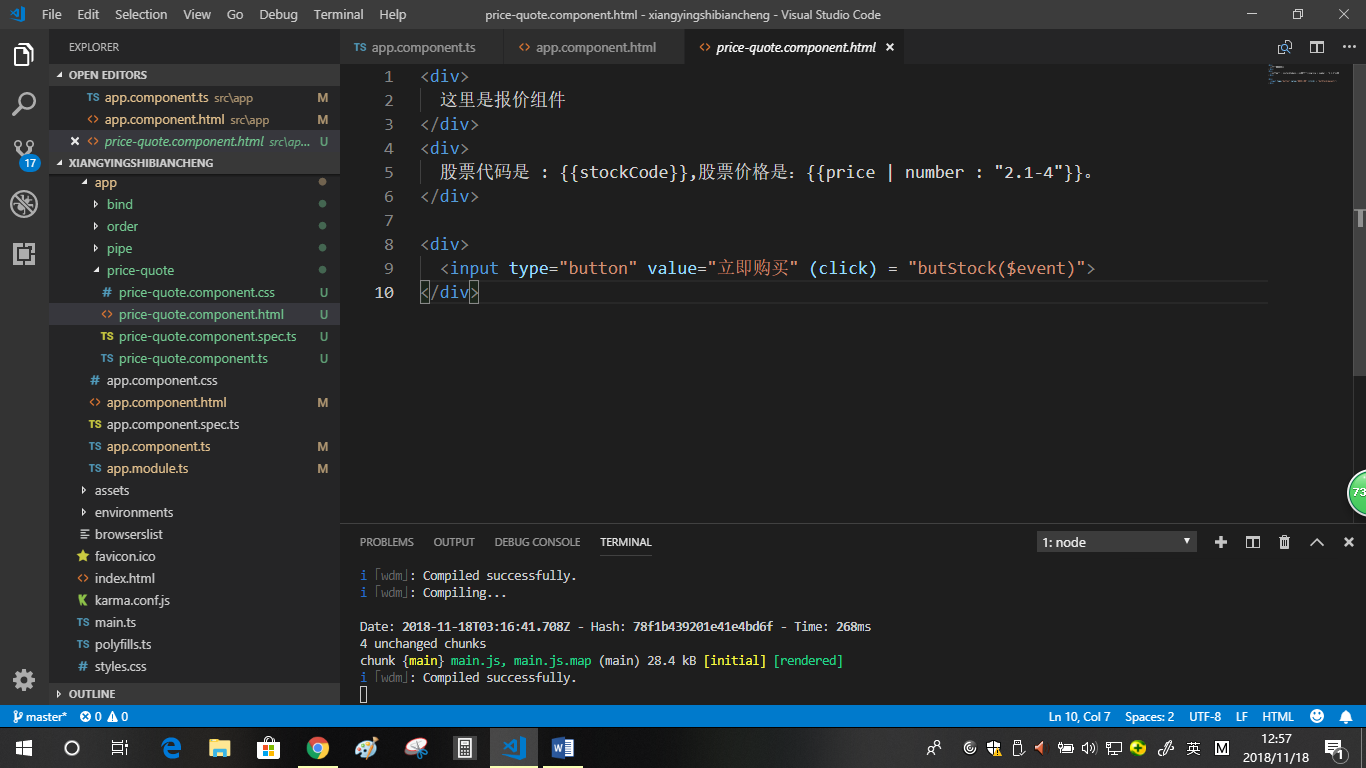
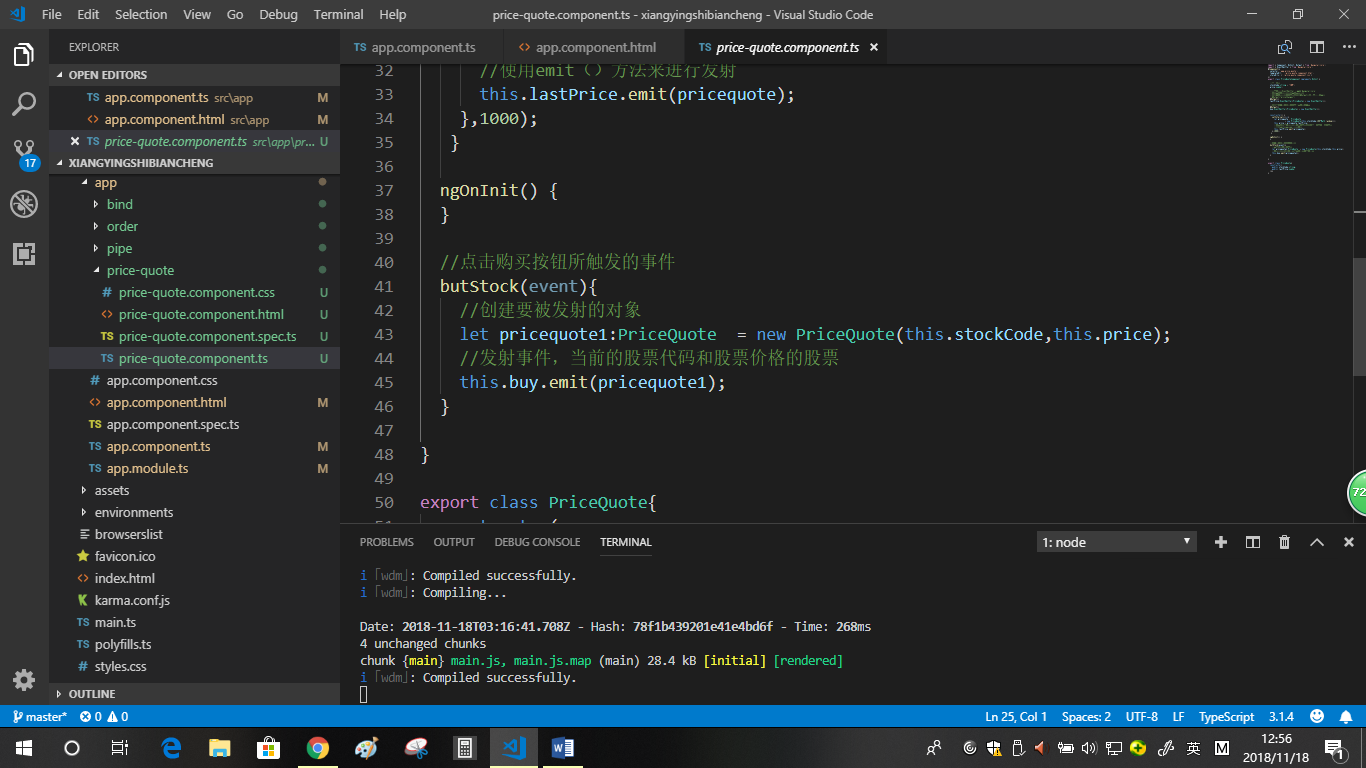
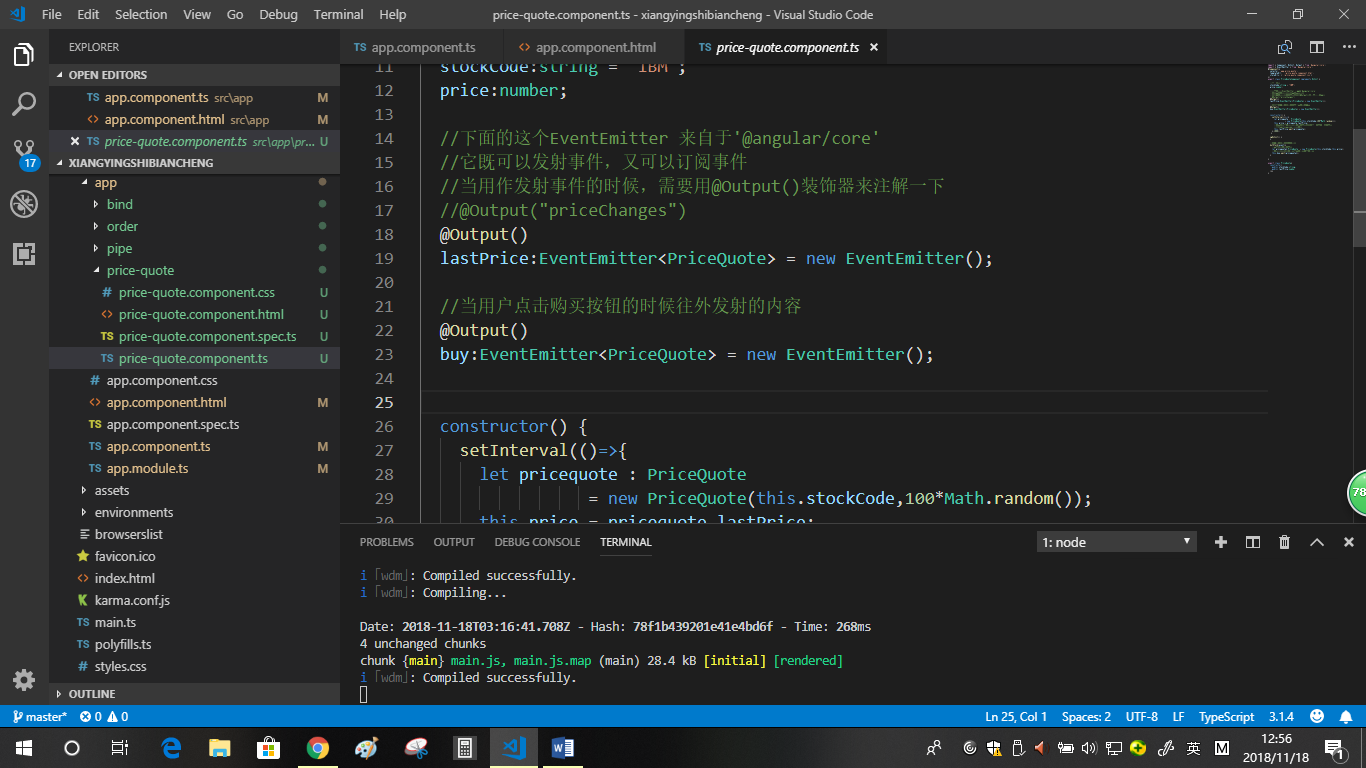
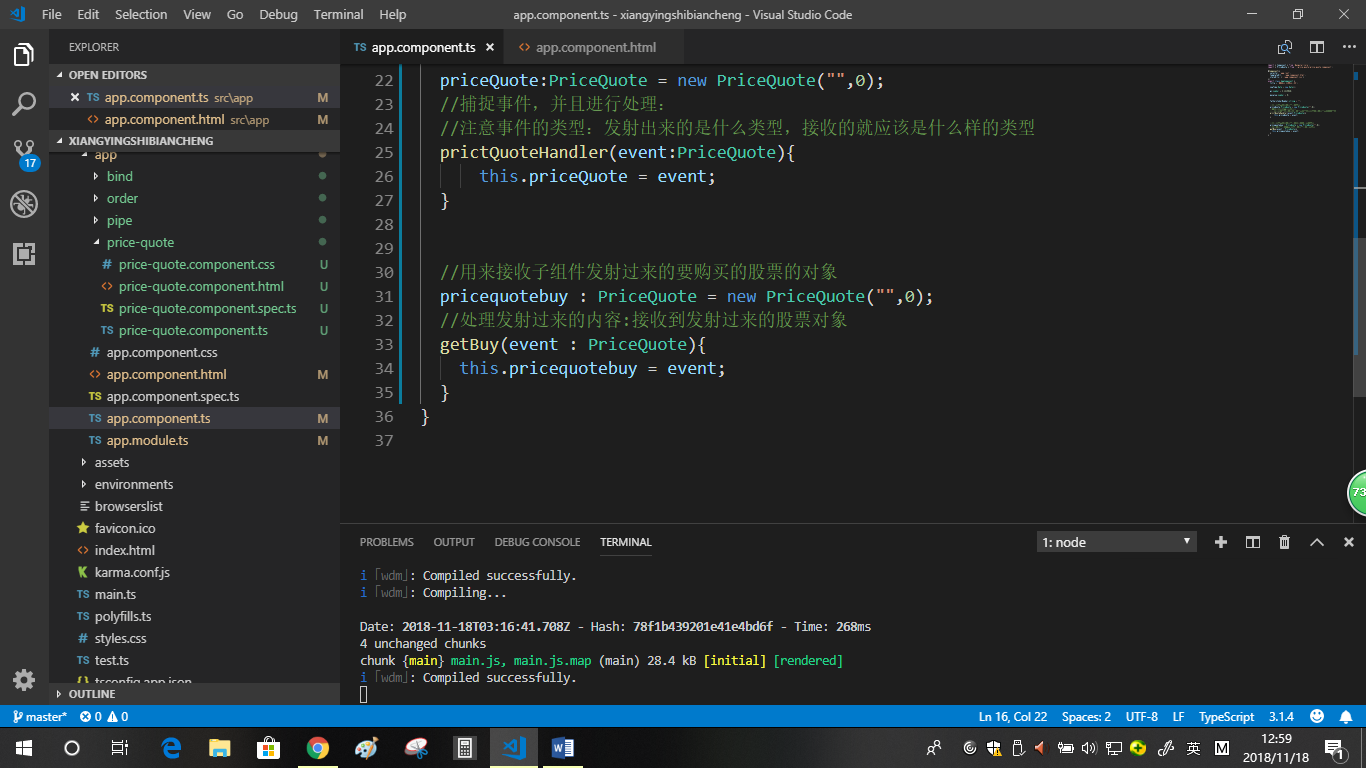
父组件

子组件1

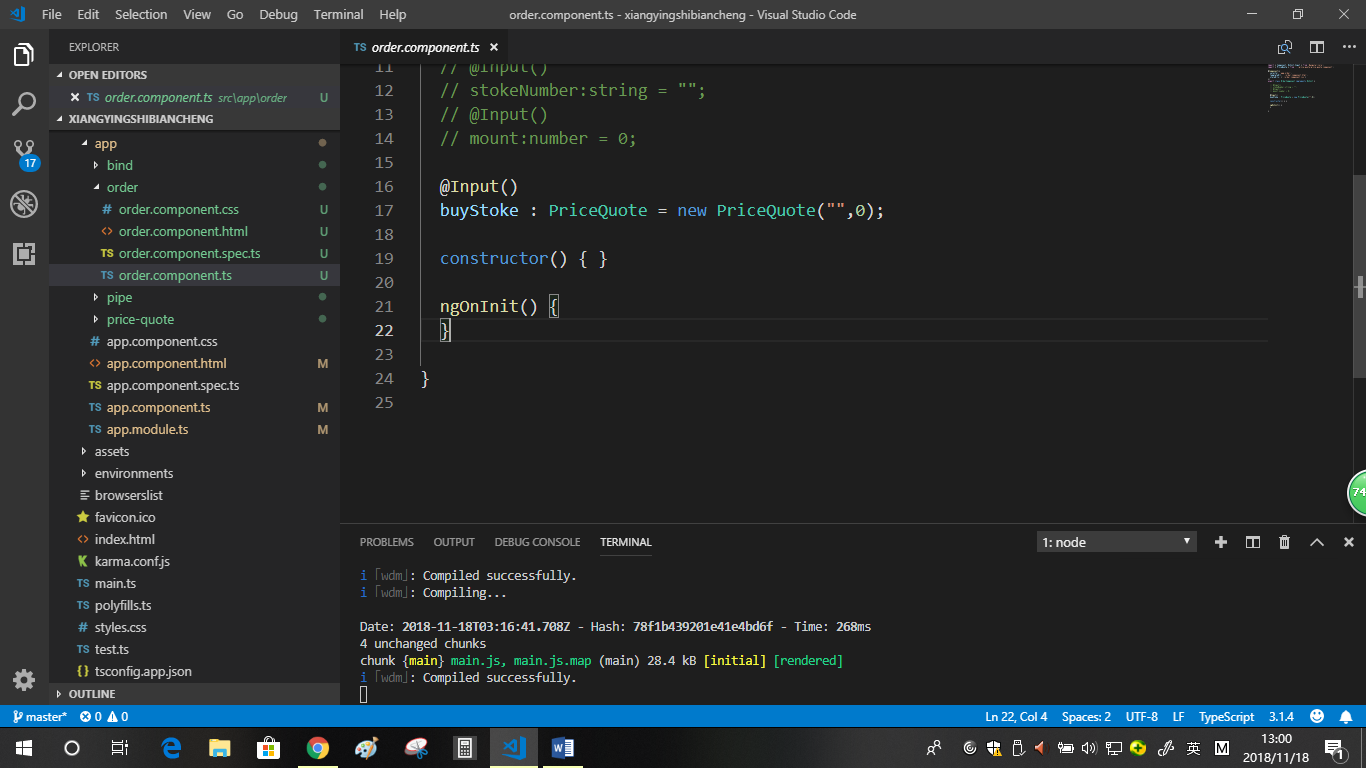
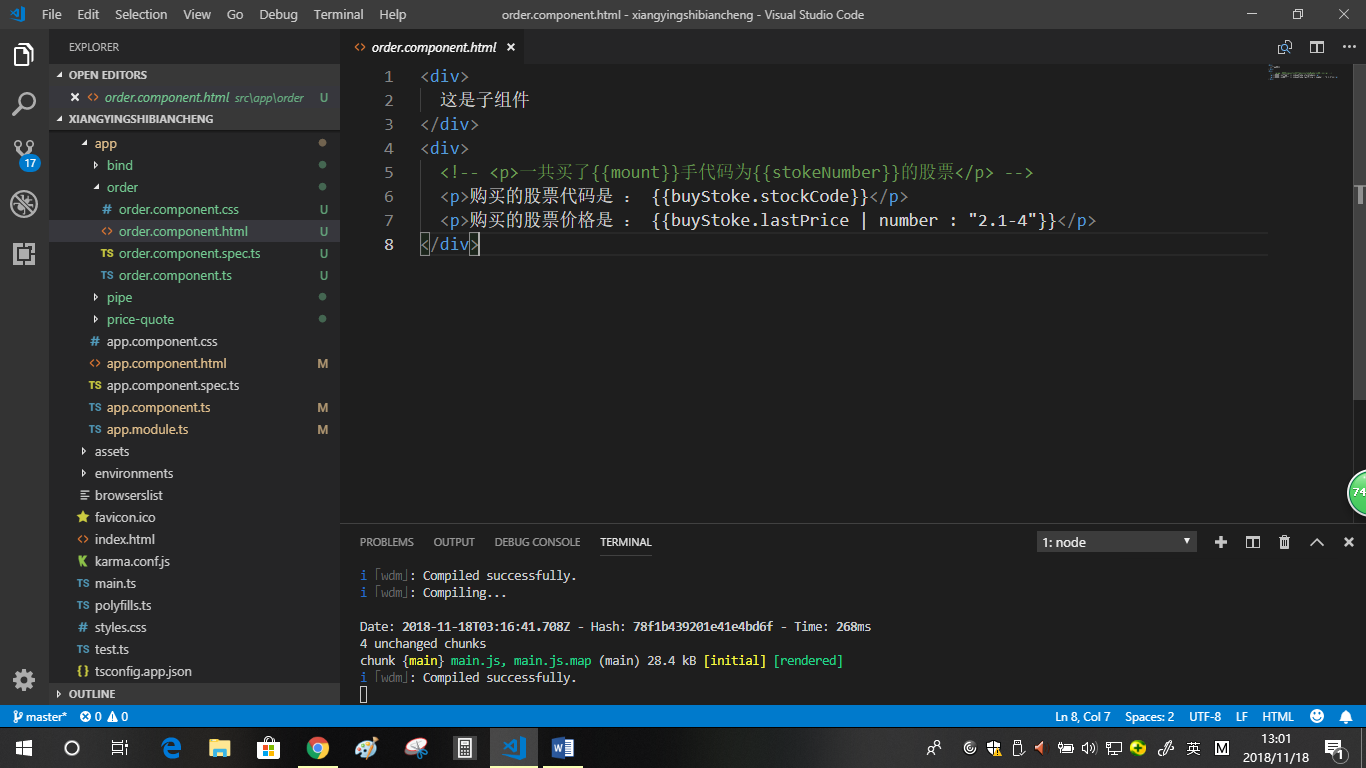
子组件2

子组件1-🡪父组件 ： 输出属性：@Output()

父组件-🡪子组件2： 输入属性：@Input()

1. 子组件1：
2. 父组件：



1. 子组件2：

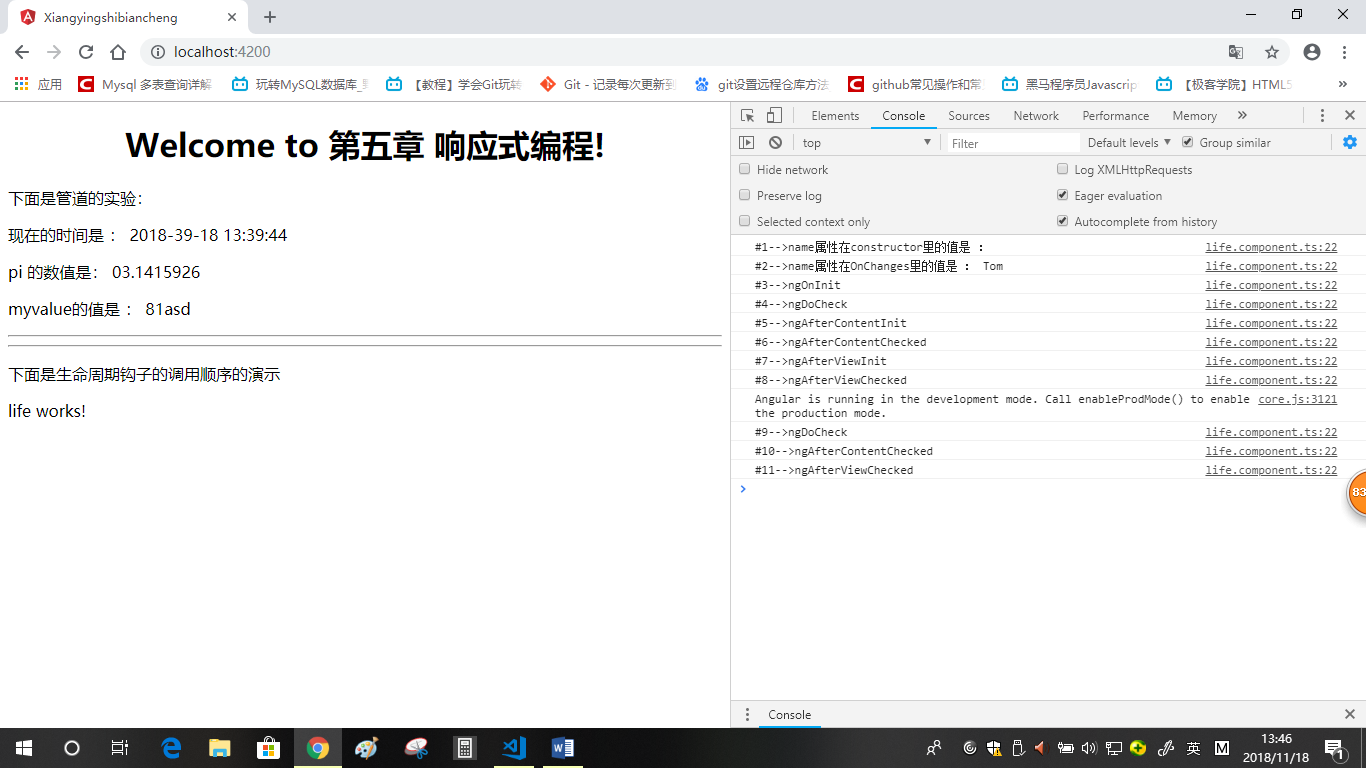
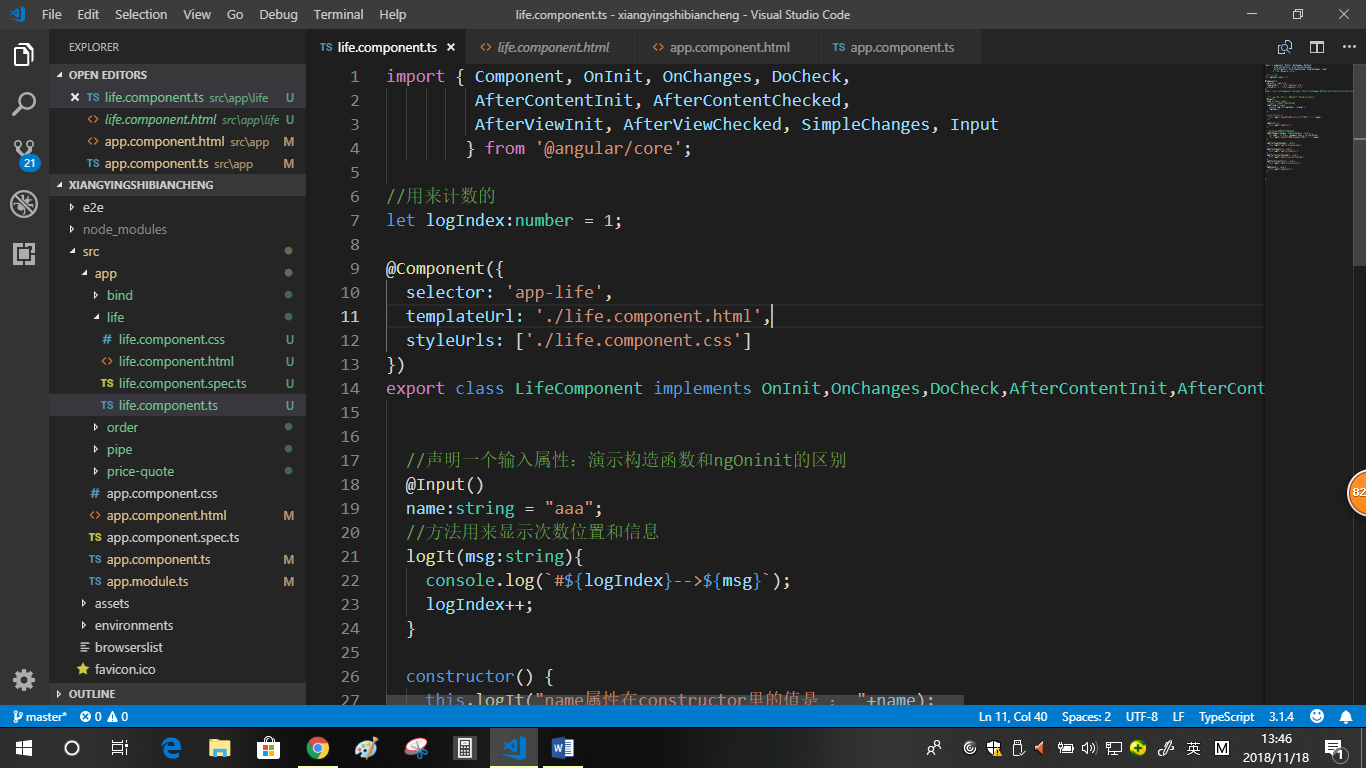


子组件1发射

运行效果图：

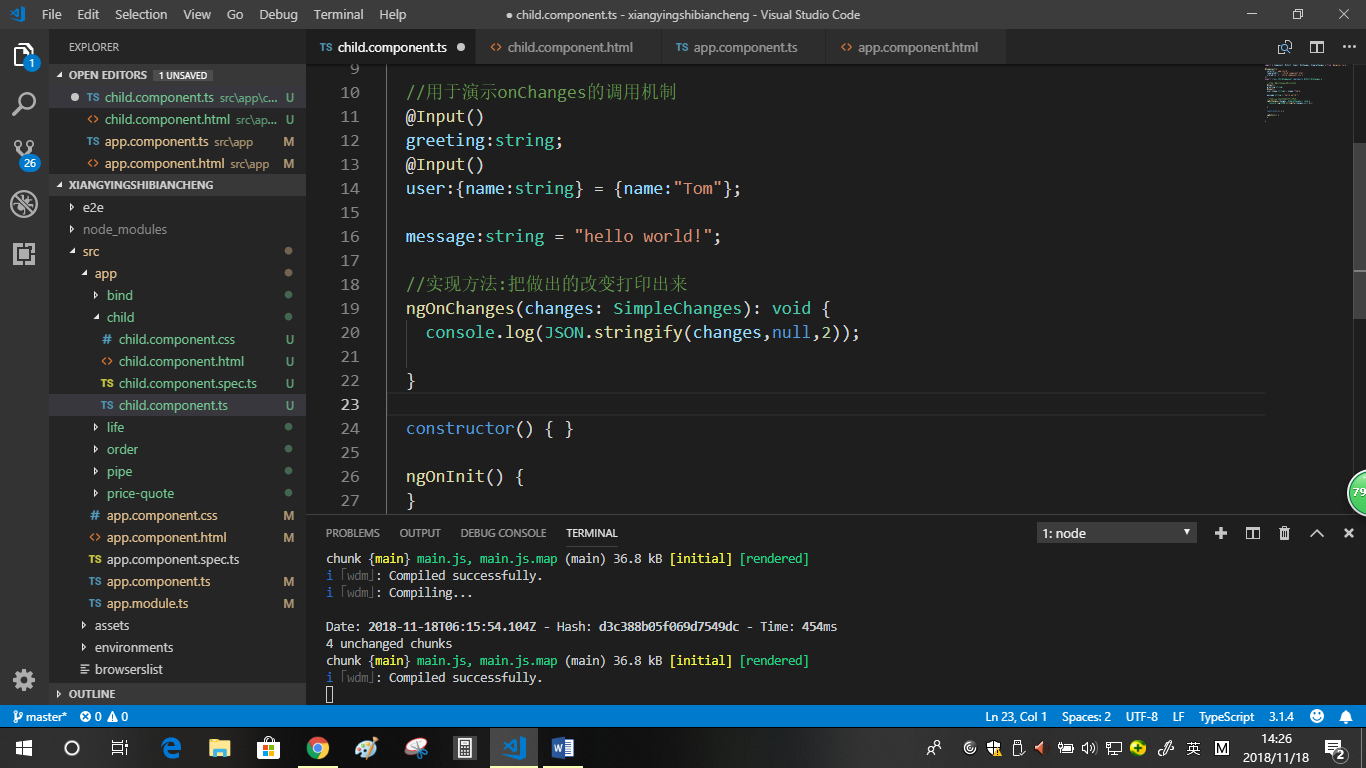
父组件接收并注入

子组件2接收

1. Angular的声明周期钩子，共9个

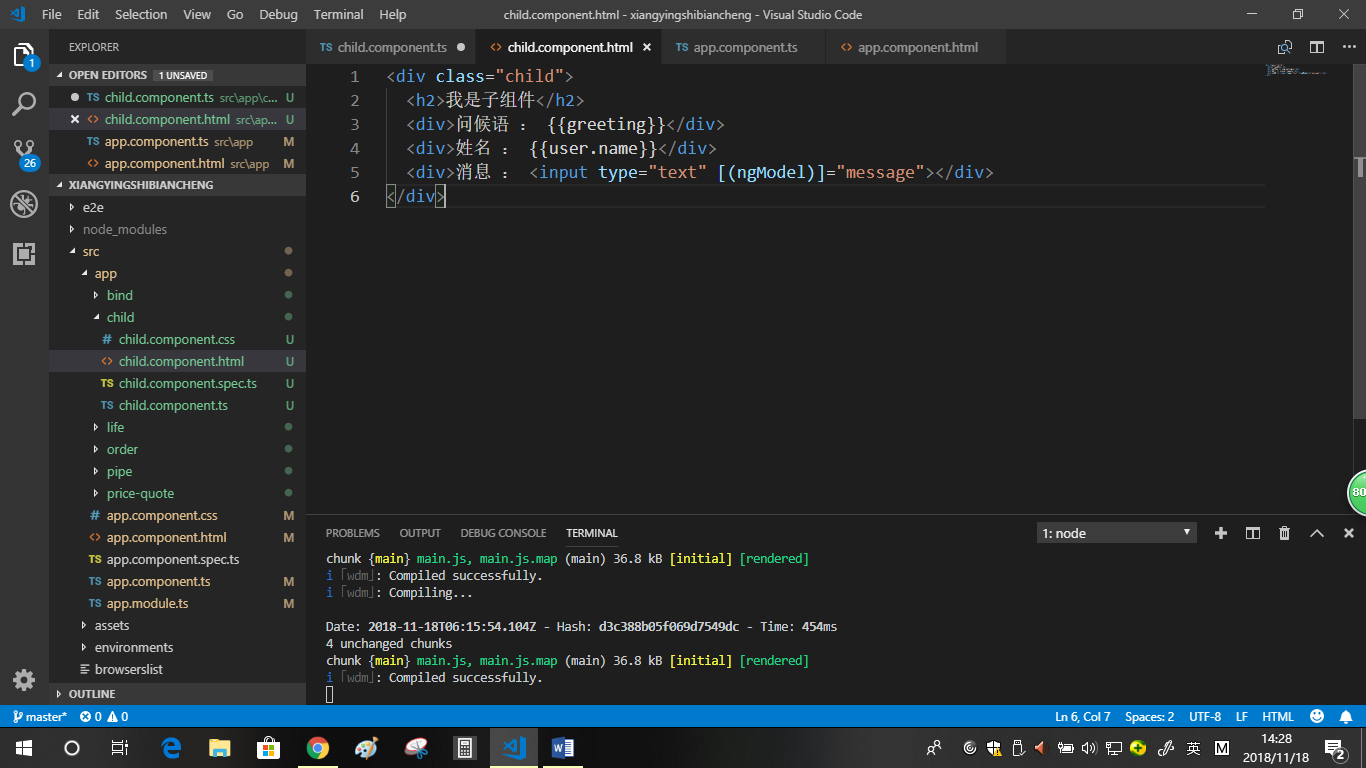
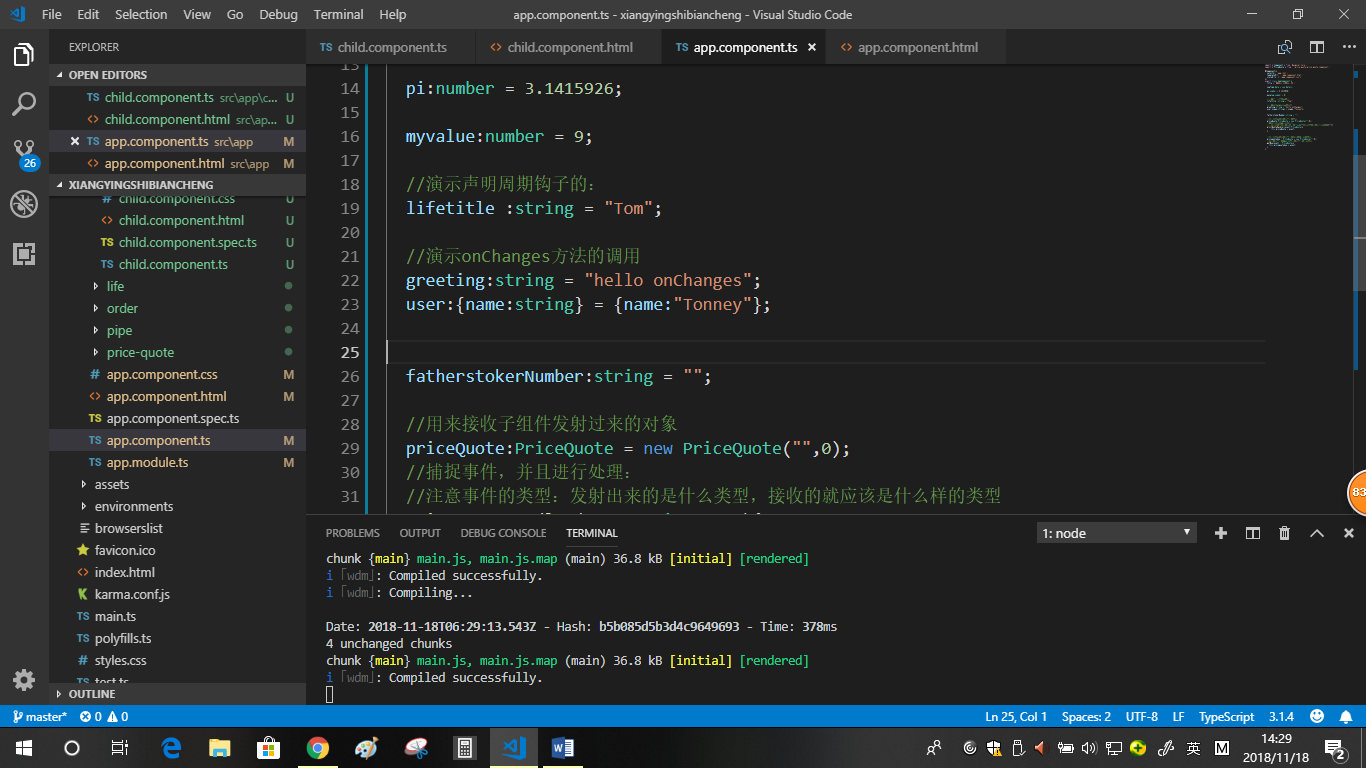
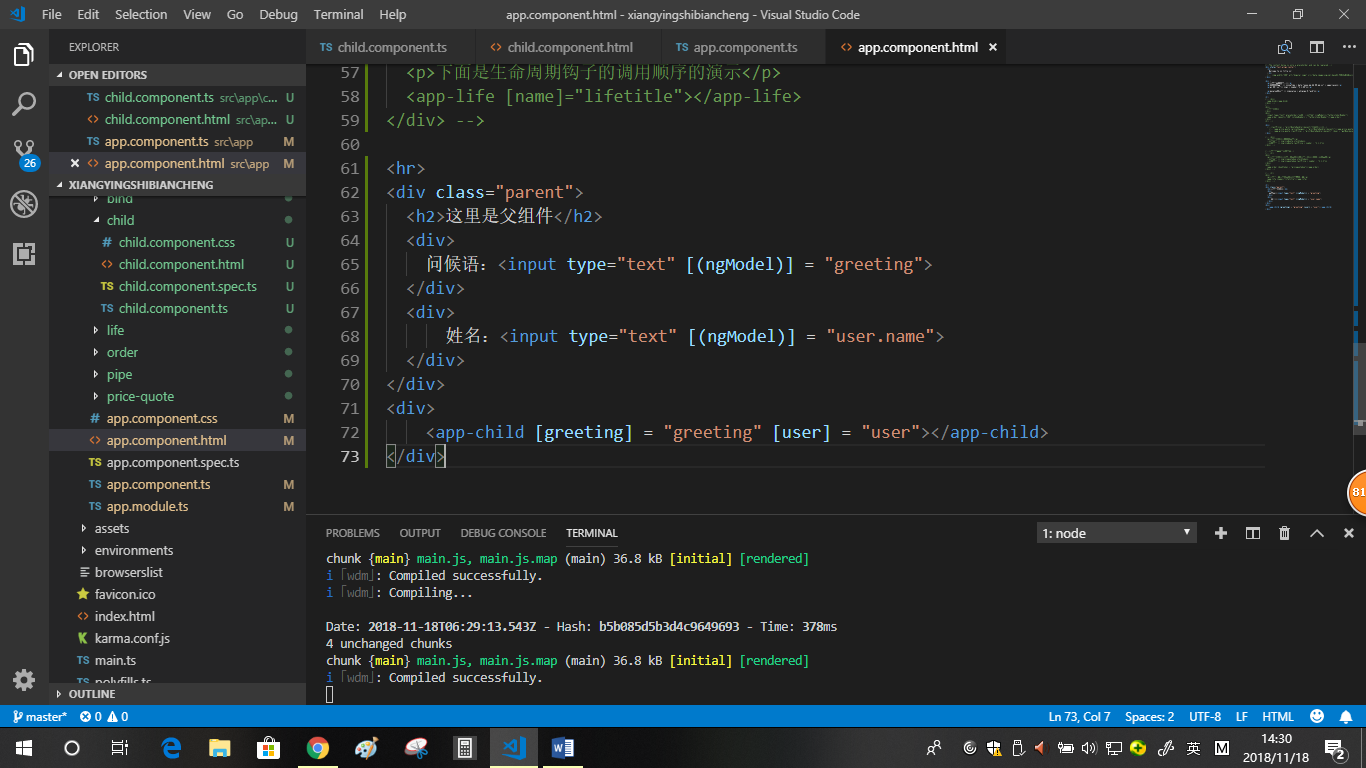
ngOnChanges():当父组件修改子组件的输入属性的时候被调用

ngOnInit():当初始化的内容以来输入属性的值的时候，初始化的逻辑要写在ngOnInit()方法中，而不要写在构造函数中

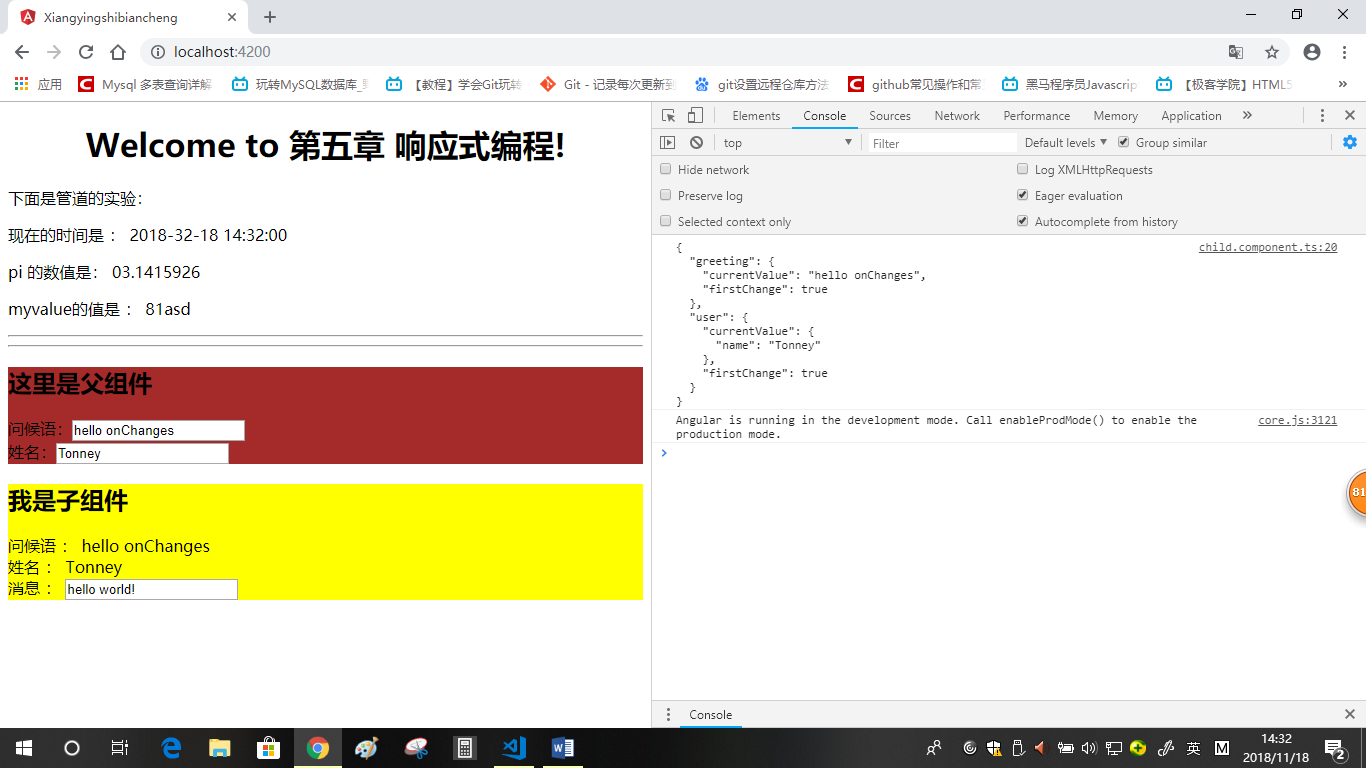
1. OnChanges钩子：
2. 子组件ts中：

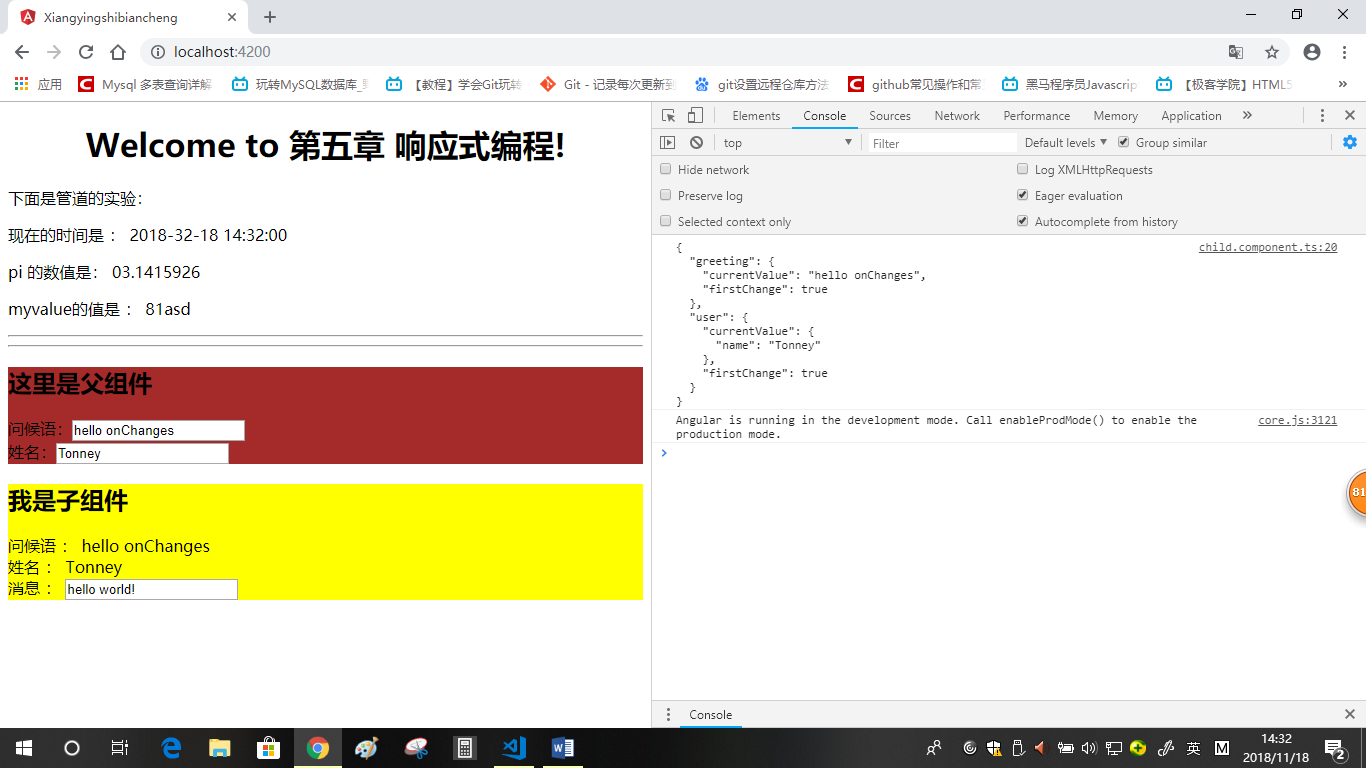
普通属性

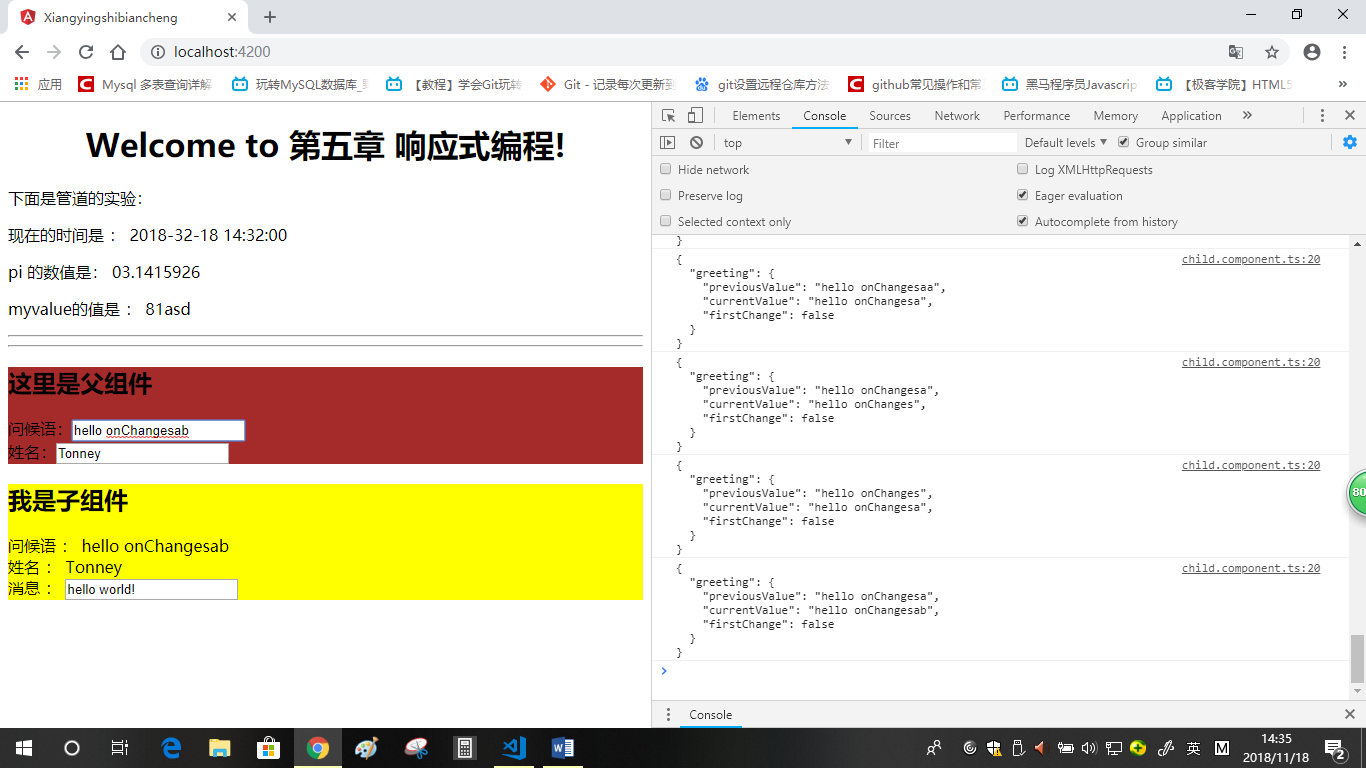
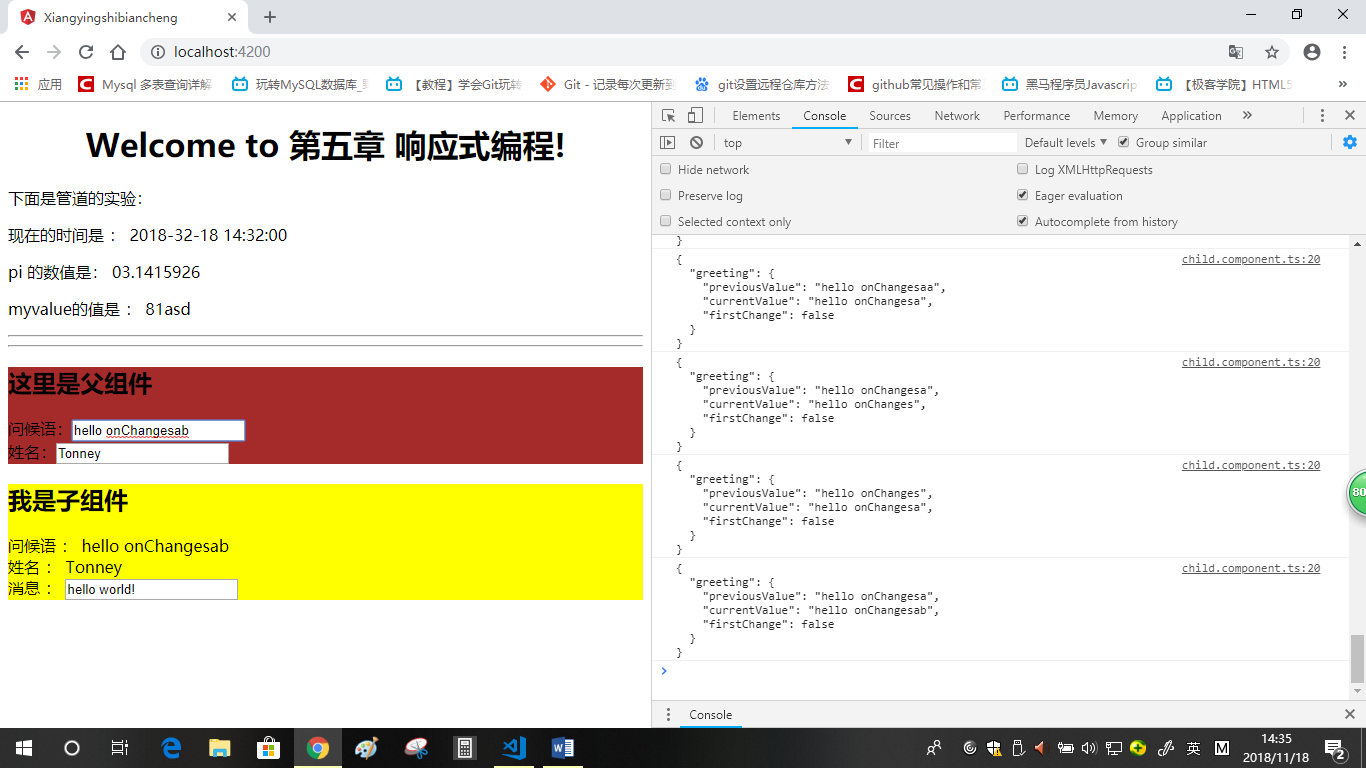
输入属性

1. 子组件html中：
2. 父组件ts中：
3. 父组件hrml中：

运行效果图：

1. 第一次运行的效果：



1. 改变父组件 问候语 ：发

@Input+不可变对象=》触发OnChanges()方法

而姓名为可变对象=》不能触发OnChanges()方法

而消息不是@Input输入属性=》不能触发OnChanges()方法