```
        })
        App.mount("#Application")
    </script>
</body>
```

运行上面的代码，可以看到页面成功渲染出了组件定义的HTML模板元素，并且可以正常地触发按钮的单击交互方法，但是我们无论怎么单击按钮，页面上渲染的数字永远不会改变。从控制台可以看出，myData对象的value属性已经发生了变化，但是页面并没有被刷新。这是由于myData对象是我们自己定义的普通JavaScript对象，其本身并没有响应式，对其进行的修改也不会同步刷新到页面上，这与我们常规使用组件的data方法返回的数据不同，data方法返回的数据会被默认保存成Proxy对象，从而获得响应式。

为了解决上面的问题，Vue 3.0中提供了reactive方法，使用这个方法对自定义的JavaScript对象进行包装，即可以方便地为其添加响应式。修改上面代码中的setup方法如下：

```
setup () {
    let myData = Vue.reactive({
        value:0
    })
    function click() {
        myData.value += 1
        console.log(myData.value)
    }
    return {
        myData,
        click
    }
}
```

再次运行代码，当myData中的value属性发生变化时，已经可以同步进行页面元素的刷新了。

7.1.3 独立的响应式值 Ref 的应用

现在，相信你已经可以熟练地定义响应式对象了。在实际开发中，很多时候我们需要的只是一个独立的原始值，以7.1.2节的示例代码为例，我们需要的只是一个数值。对于这种场景，不需要手动将其包装为对象的属性，可以直接使用Vue提供的ref方法来定义响应式独立值，ref方法会帮我们完成对象的包装，示例代码如下：

```
<script>
    const App = Vue.createApp({
        setup () {
            // 定义响应式独立值
            let myObject = Vue.ref(0)
            // 需要注意，myObject会自动包装对象，其中定义value属性为原始值
            function click() {
                myObject.value += 1
                console.log(myObject.value)
```