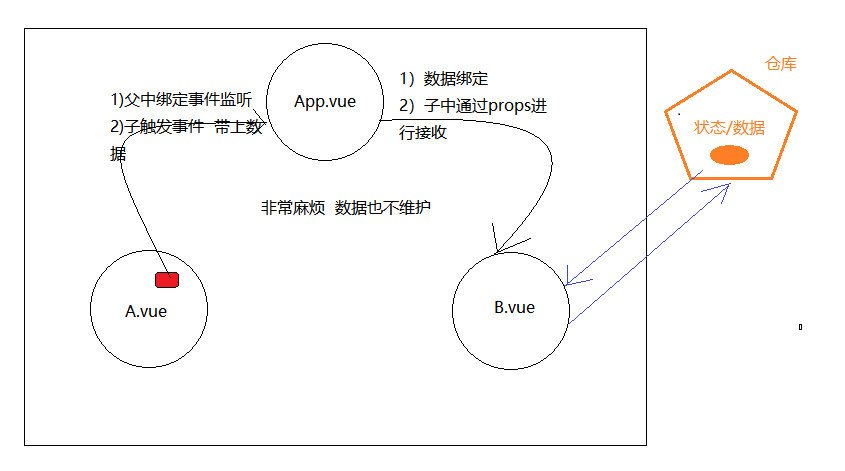
# Vuex（难一点）

Vue应用程序的**状态(**state**)管理器。**它采用集中式存储管理应用的所有组件的状态，并以相应的规则保证状态以一种可预测的方式发生变化。



把“状态”换成“数据”。

把vue应用的数据放在一个地方管理。 每一个vue的组件都可以有data项（有自己的数据），我们可以通过父子之间的相互传递，进行数据的交换。

父传子：自定义的属性; 子组件用props

子传父：自定义的事件；子组件用emit

## 为什么要用它

如果你希望在子组件与另一个组件之间共享（它们是兄弟，表兄弟，堂兄弟关系）数据，则非常麻烦。

如果你的项目中，需要用到在各个子组件中共享数据，则你就需要用到vuex。

## 示例

**设置如下组件：**

* 一个vue实例，充当根组件
* 子组件AddNumber。
* 子组件SubNumber。

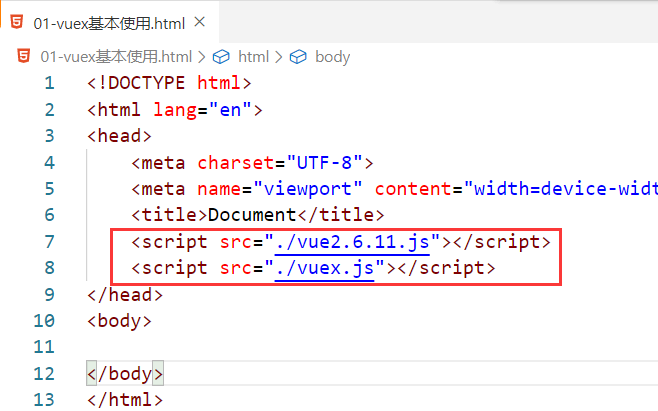
**希望在两个子组件AddNumber和subNumer之间共同维护和使用数据项:counter**

* 在vue实例中：显示 counter的值
* 在子组件AddNumber中：显示 counter的值；添加counter的值
* 在子组件subNumber中：显示 counter的值；减少counter的值

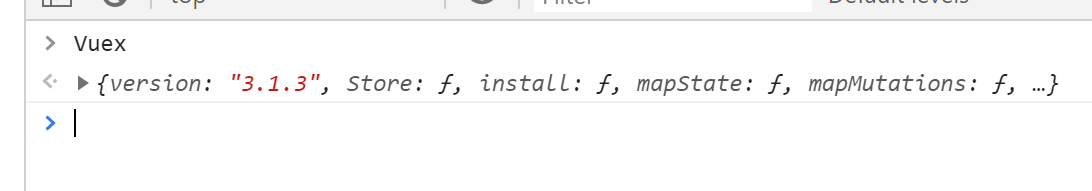
由于上面的需要，我们需要在三个地方共同使用同一数据项counter，所以我们需要用到 vuex。

## 步骤

### 先引入vue.js，再引入vuex.js



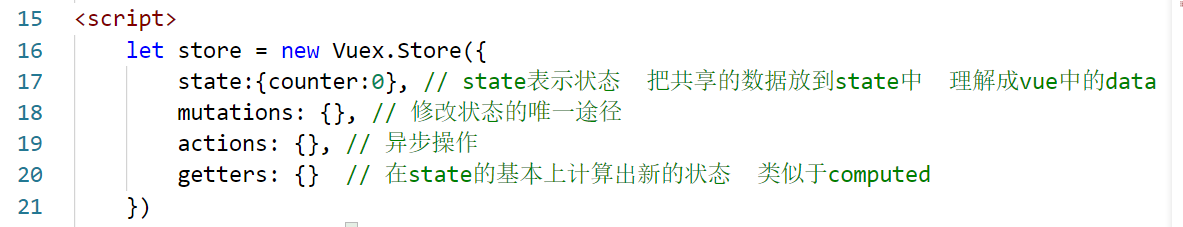
此时，你已经可以在控制台中访问vuex



### 实例化vuex中的**store**对象

实例化一个对象，就是通过new的方式去创建一个对象。

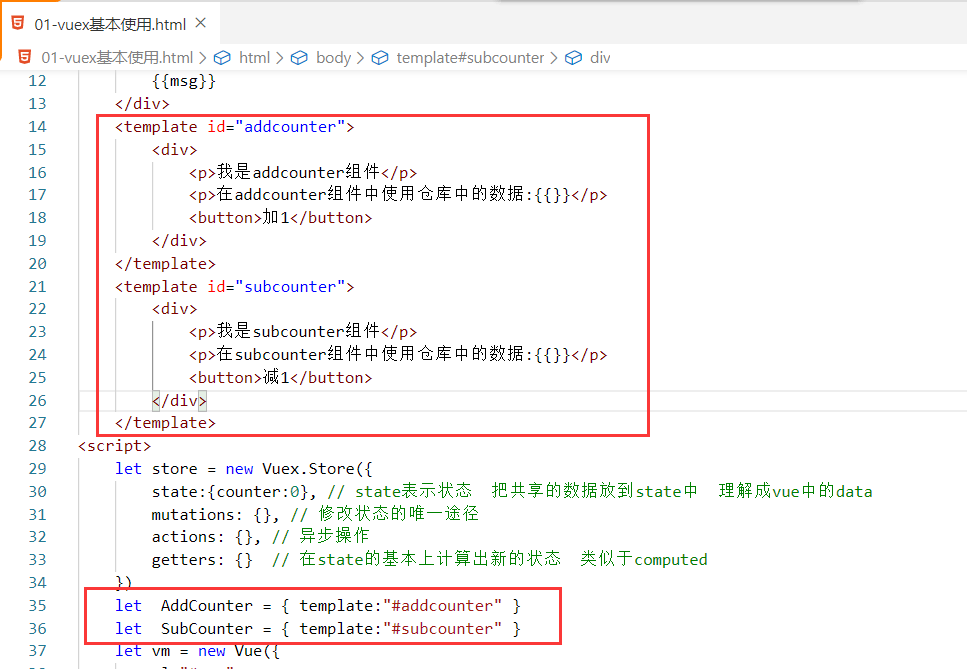
#### 格式



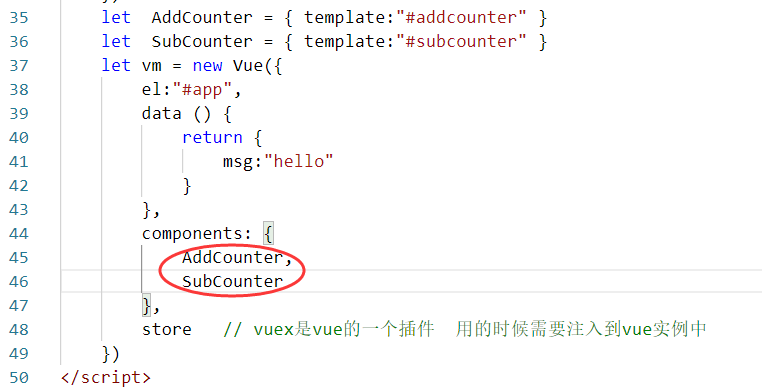
### 注入到Vue实例中



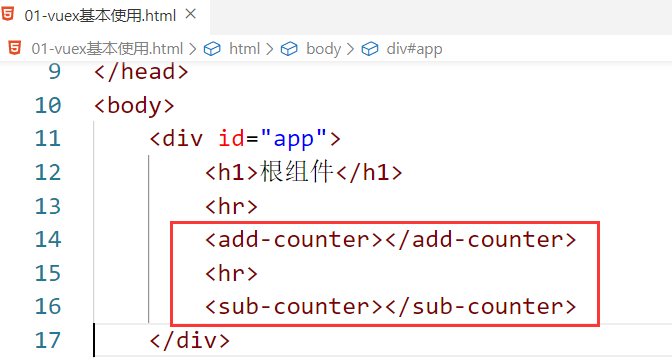
定义两个组件如下：



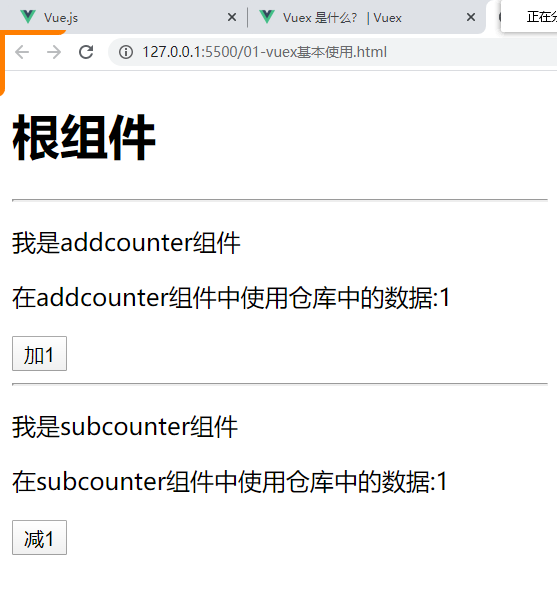
不要忘了，组件需要注册，如下：



使用组件，如下：



效果如下：



### 使用store中的数据

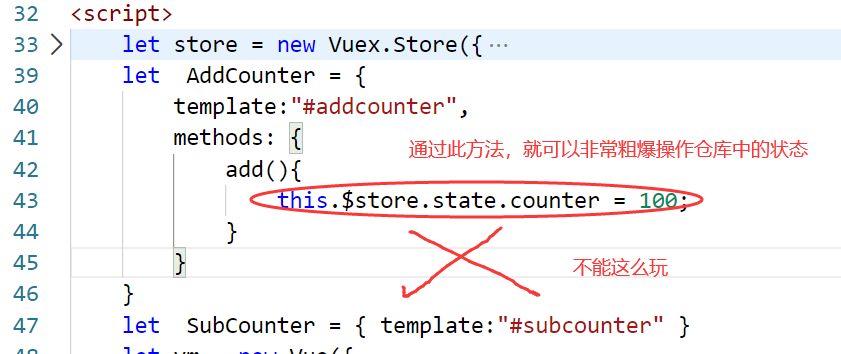
一旦你在vue的实例中注入了store，则在所有的子组件及 vue的实例中，你都可以通过：this.$store.state.数据名 去获取数据

类似于我们把router注入到vue实例中，我们就可以通过this.$router 和 this.$route操作路由。

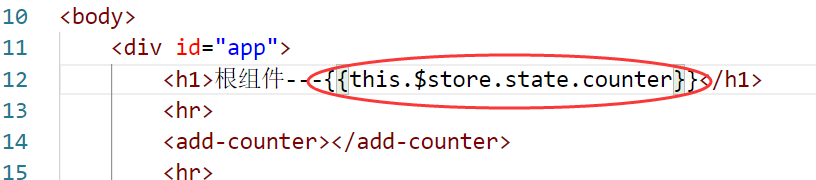
在组件中使用仓库中的数据，如下：



也可以通过：this.$store.state.数据名 = 值 去修改数据，但是，vuex反对这么做。



**你可以有两种方法去使用数据：**

1. 获取：this.$store.state.数据名  
   
2. 修改：在组件内部通过this.$commit方法触发事件，执行mutations当中的对应的方法。

## 最核心的概念- store

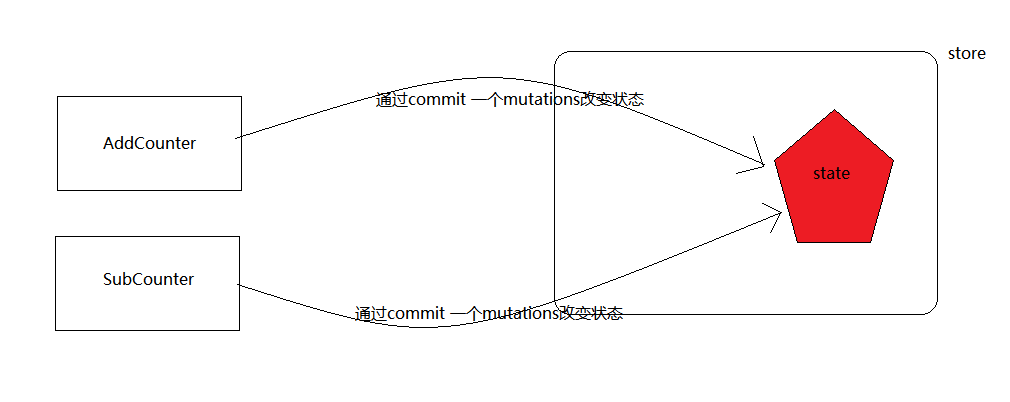
每一个 Vuex 应用的核心就是 store（仓库）。

store是一个容器，包含着vue应用的状态(state)。

有如下两大特点：

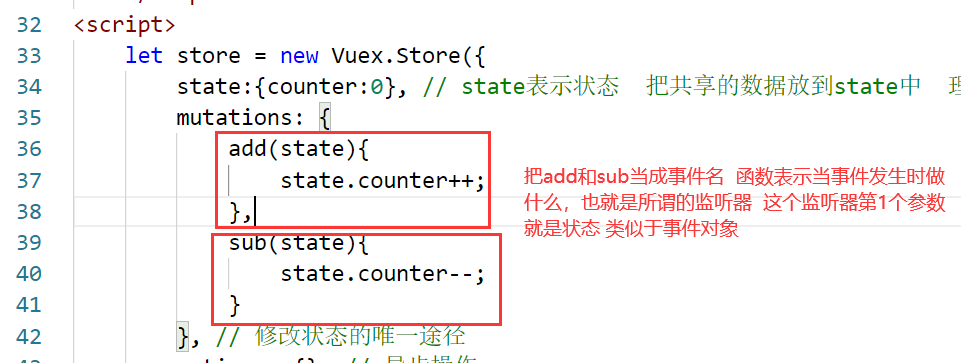
Vuex 的状态存储是响应式的。当 Vue 组件从 store 中读取状态的时候，若 store 中的状态发生变化，那么相应的组件也会相应地得到高效更新。

你不能直接更改 store 中的状态。**改变** store 中的状态的**唯一途径就是显式地提交(commit) mutations**。这样使得我们可以方便地跟踪每一个状态的变化.



## Mutations

### 定义格式：



### 调用这个mutations

它是间接被调用的，在组件内部：

this.$store.commit(“mutations中的属性名”) . 代码如下：

