## 第十二天学习内容

 Vuex的介绍

* + Vuex 是一个专为 Vue.js 应用程序开发的状态管理模式。
  + 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。
  + Vuex用来解决多个组件数据传输的问题

 状态管理模式

VueX中的核心内容

* + state 存放状态
  + mutations state成员操作
  + getters 加工state成员给外界
  + actions 异步操作
  + modules 模块化状态管理

 Vuex的选择

* + 优点：帮助我们管理共享状态，并附带了更多的概念和框架。这需要对短期和长期效益进行权衡。
  + 缺点：不打算开发大型单页应用，使用 Vuex 可能是繁琐冗余的。
  + 简单的页面开发推荐使用store模式，但是，如果您需要构建一个中大型单页应用，您很可能会考虑如何更好地在组件外部管理状态，Vuex 将会成为自然而然的选择。

 安装Vuex把项目配置到Vuex

* + 安装vuex依赖包

npm install vuex --save

* + 导入vuex包

import Vuewx from 'vuex'

vue.use(Vuex)

* 1. 创建store对象

const store = new Vuex.Store({

State:{ count: 0 }

})

* 1. 在vue实例中挂载store

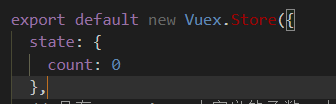
 创建一个含有vuex的项目

创建两个组件，实现两个组件数据的传输

 学习State的概念与使用

 Vuex 使用单一状态树

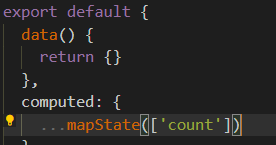
* + 提供唯一的公共数据源，所有的共享数据都放在state中
  + Vuex 的状态存储是响应式的。
  + 可以直接定位任一特定的状态片段，调试过程中能轻易地取得整个当前应用状态的快照。
  + 单状态树和模块化并不冲突。
  + 存储在 Vuex 中的数据和 Vue 实例中的 data 遵循相同的规则。
  + 从Vue 组件中获得 Vuex 状态。



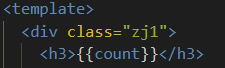
* + 组件访问state数据的两种方式
    1. 在模板中通过{{$store.state.count}}访问数据
    2. 使用mapState访问,在组件中使用
       1. 在组件中使用vuex中导入mapState



* + 将全局数据映射到当前组件，展开运算符…

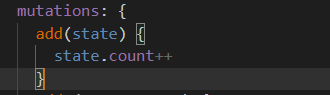


* 1. 将数组里的参数加入模板中进行使用



 Mutation的概念与使用

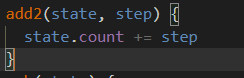
* + 使用mutation变更store里的数据
  + mutation监控着store的数据
  + mutation中的函数不能定义**异步**
  + 使用mutation
    - 在moution中定义对数据的操作方法



* + 在组件中调用roater中的mutations中的方法

this.$store.commit('add')

* + 使用mutation时组件传递参数
    - 在mutation中定义的函数传递多个参数



* + 在组件中传值

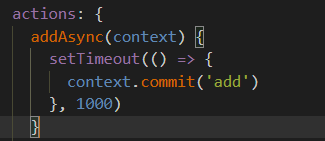
this.$store.commit('add2', 2)

* + 在调用mutation的定义方法时第二种方法mapMutations函数（与mapState用法相似）
    - 引入mapMutations
    - ...mapMutations(['sub', 'sub2'])
    - 在方法中调用

 使用了mapMutations引入的全局函数，在组件中的方法中用this就可以随意使用

 Action的概念与使用

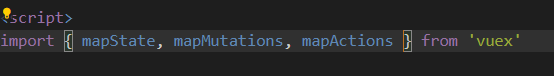
* + mutation中不能使用异步操作，所以提供action来实现异步
  + 因为只有mutation能修改state的数据，所以action异步中要调用mutation的函数修改state里的数据
  + 触发action的两种方式
    - 使用dispatch
      1. 使用context.commit('方法名')来调用



* + 在组件中使用异步函数方法，dispatch用来触发action

this.$store.dispatch('addAsync')

* 1. 使用mapActions方法调用
     1. 在组件中使用vuex中导入mapAction



* 1. 在当前组件的方法中映射

...mapActions(['subAsync']),

* 1. 在组件中使用

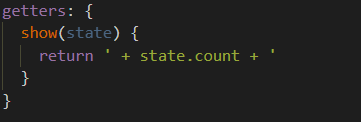
直接可以作为方法使用

 Module

* + 随着项目的复杂度增大，为了方便管理vuex
  + 可以将 store 分割成模块（module）。每个模块拥有自己的 state、mutation、action、getter、甚至是嵌套子模块——从上至下进行同样方式的分割

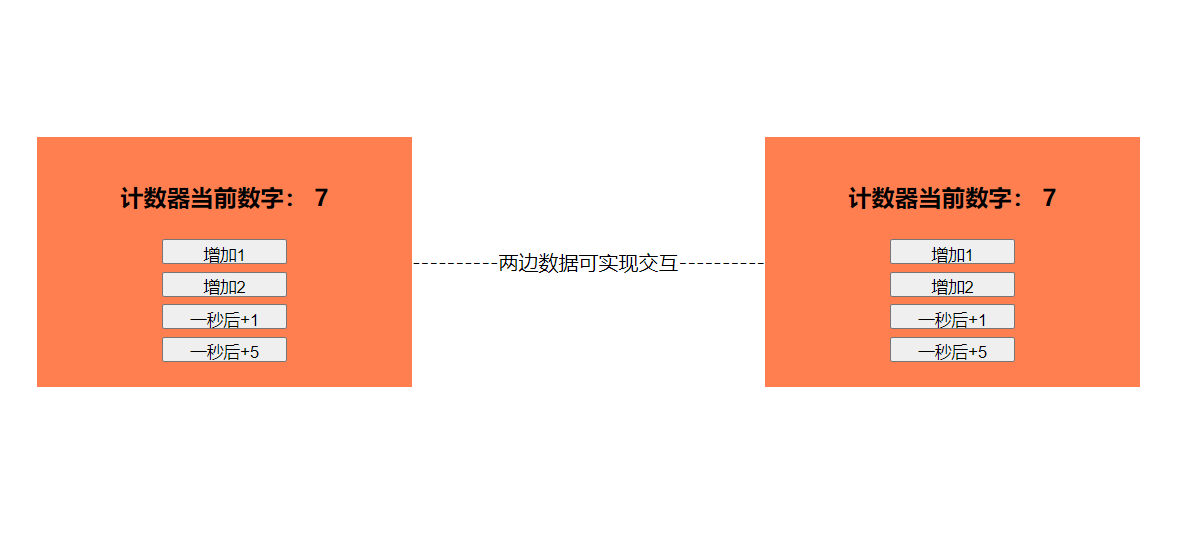
 Getter的概念与使用

* + getter不会修改state的数据，可以包装数据
  + 定义getters



* + 使用getters两种方法
    1. $store.getters.show
    2. 使用maoGetter

 运用今天知识参考b站视频制作两个组件计数器，两个计数器的数值通过store进行连接



 将今天的计数器部署在之前的vue项目中

加入vue中无法编译，报错。（继续解决）

 使用webpack打包

 上传呢github

github地址：https://github.com/Kang-kang666/study\_2020-10-28