# Vuex 概述

## 组件之间共享数据的方式

1．父向子传值：v-bind 属性绑定

2．子向父传值：v-on 事件绑定

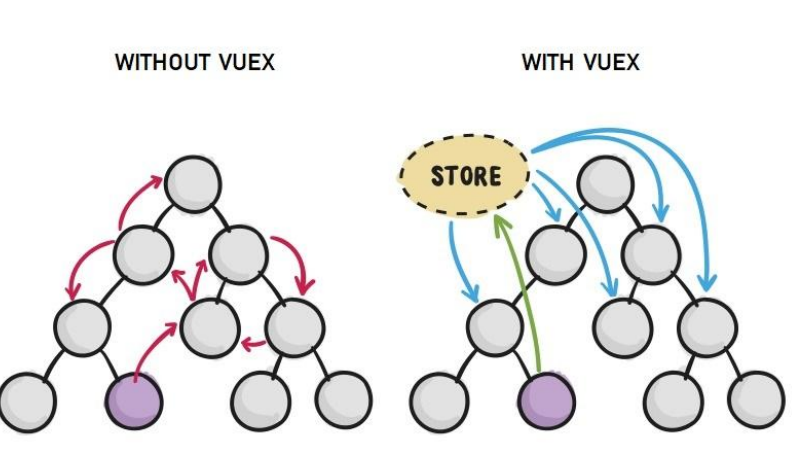
3．兄弟组件之间共享数据： EventBus 

**$on 接收数据**的那个组件 

**$emit 发送数据**的那个组件

## 1.2 Vuex 是什么

Vuex 是实现**组件全局状态（数据）管理**的一种机制，可以方便的**实现组件之间数据的共享**。



## 1.3 使用 Vuex 统一管理状态的好处

①能够在 vuex 中集中管理共享的数据，易于开发和后期维护

②能够高效地实现组件之间的数据共享，提高开发效率

③存储在 vuex 中的数据都是响应式的，能够实时保持数据与页面的同步

一般情况下，只有组件之间共享的数据，才有必要存储到 vuex 中；对于组件中的私有数据，依旧存储在组件自身的 data 中即可。

# Vuex 的基本使用

1. 安装 vuex 依赖包

npm install vuex --save

2. 导入 vuex 包

import Vuex from 'vuex'

Vue.use(Vuex)

3. 创建 store 对象

const store = new Vuex.Store({

// state 中存放的就是全局共享的数据

state: { count: 0 }

})

4. 将 store 对象挂载到 vue 实例中

new Vue({

el: '#app',

render: h => h(app),

router,

// 将创建的共享数据对象，挂载到 Vue 实例中

// 所有的组件，就可以直接从 store 中获取全局的数据了

store

})

# 3. Vuex 的核心概念

## 3.1 核心概念概述 

Vuex 中的主要核心概念如下：

**State **

**Mutation**

**Action **

**Getter**

## 3.2 state---存储数据



new Vue({

  // 将创建的共享数据对象，挂载到 Vue 实例中

  // 所有的组件，就可以直接从 store 中获取全局的数据了

  store,

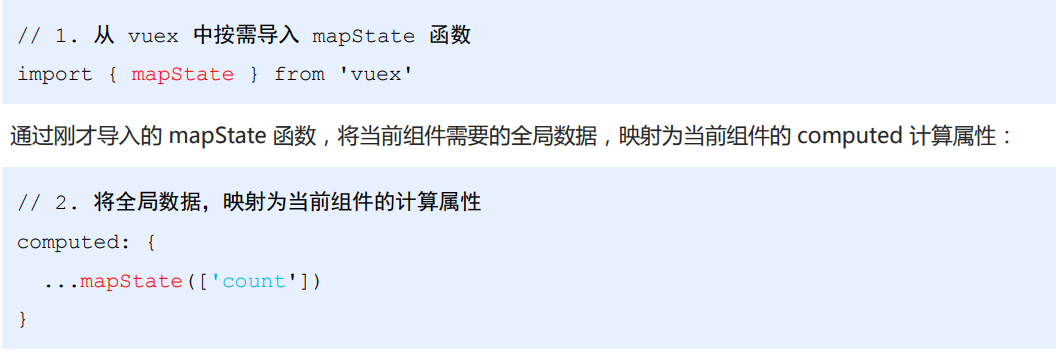
  render: (h) => h(App)

}).$mount('#app')

组件访问 State 中数据的第一种方式：



组件访问 State 中数据的第二种方式：



如：

<template>

  <div>

   <h1>查看结果: {{ count }} </h1>

   <button>-</button>

  </div>

</template>

<script>

import { mapState } from 'vuex'

export default {

  data () {

    return {}

  },

  computed: {

    ...mapState(['count'])

  }

}

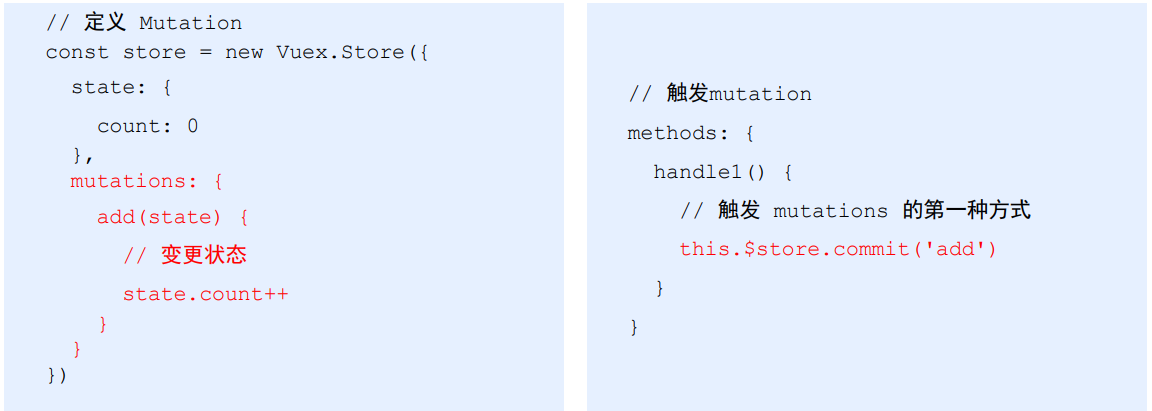
</script>

## 3.3 Mutation----变更数据

1.Mutation 用于变更 Store中 的数据。

（1）只能通过 mutation 变更 Store 数据，不可以直接操作 Store 中的数据。

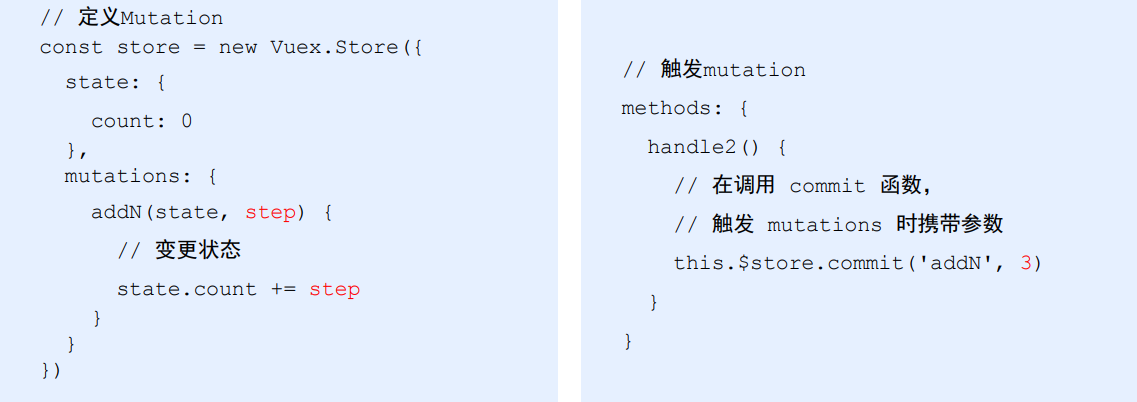
（2）通过这种方式虽然操作起来稍微繁琐一些，但是可以集中监控所有数据的变化。



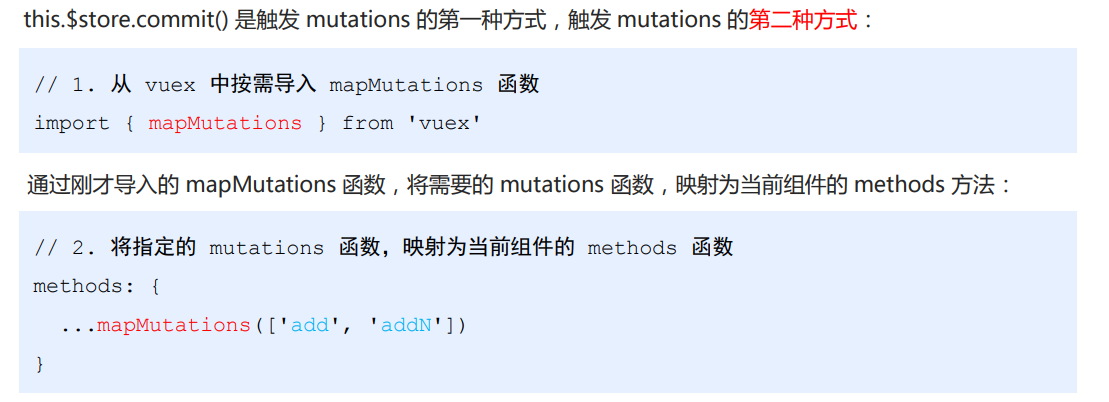
2.可以在触发 mutations 时传递参数：

（1）触发mutations的第一种方式

Commit函数



（2）触发mutations的第二种方式



**注：**mutations函数中不能写异步代码

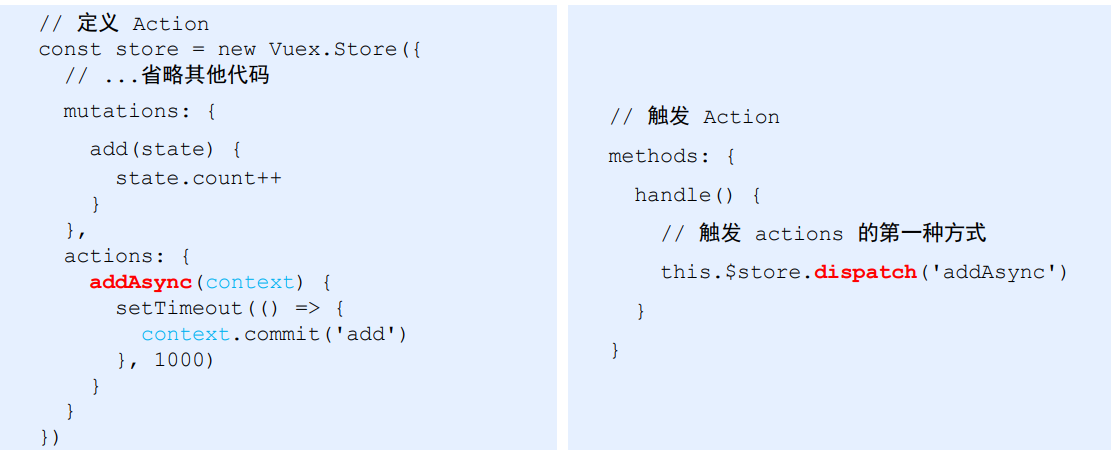
## 3.4 Action ---处理异步任务

如果通过**异步操作变更数据**，**必须通过 Action**，而不能使用 Mutation，但是在 Action 中还是要**通过触发Mutation 的方式间接变更数据**。

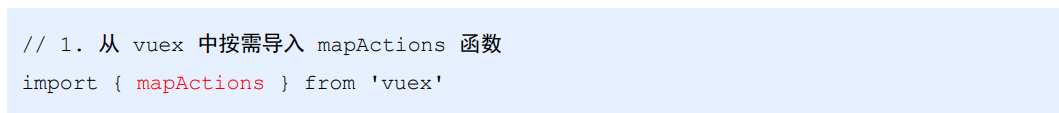
1. this.$store.dispatch() 是触发 actions 的第一种方式

 // 触发actions

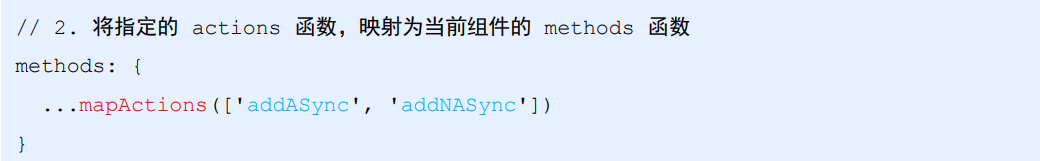
dispatch函数



2.触发 actions 的第二种方式：



通过刚才导入的 mapActions 函数，将需要的 actions 函数，映射为当前组件的 methods 方法：



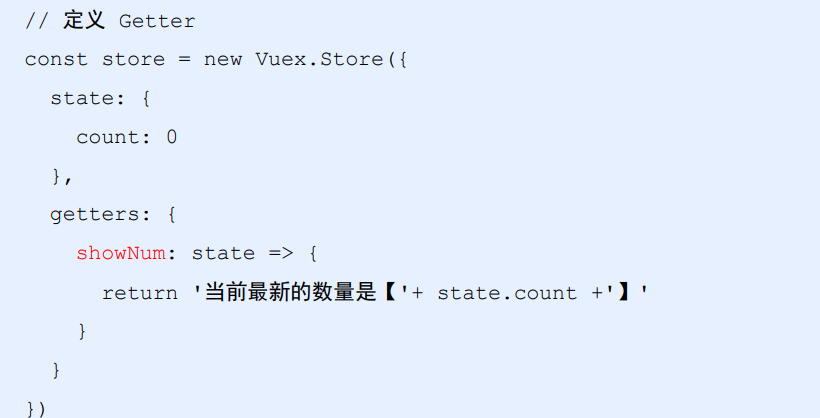
（3）.

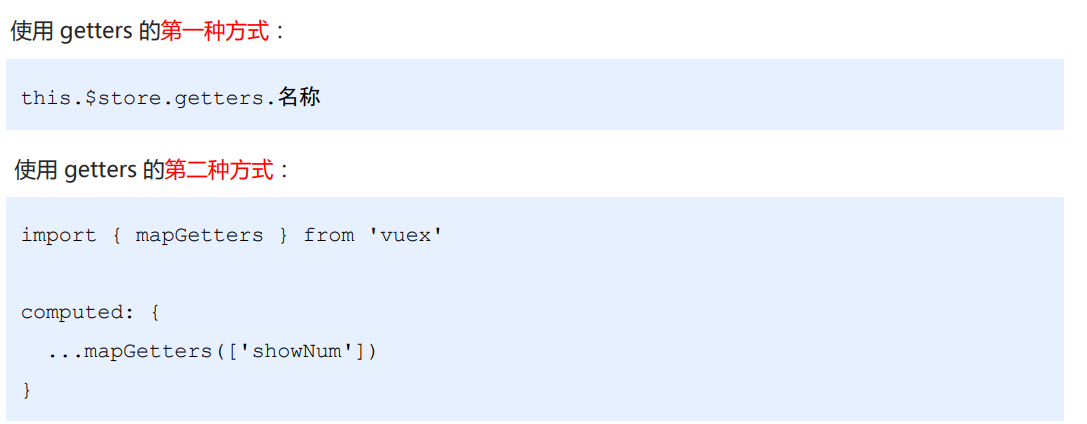
 <button @click="subNAsync(5)">-N subNAsync</button>

## 3.5 Getter—包装的作用--加工处理形成新的数据

Getter 用于对 Store 中的数据进行加工处理形成新的数据。

①Getter 可以对 Store 中已有的数据加工处理之后形成新的数据，类似 Vue 的计算属性。 ②Store 中数据发生变化，Getter 的数据也会跟着变化。





# 4．Vuex案例

1. 初始化项目

实现步骤

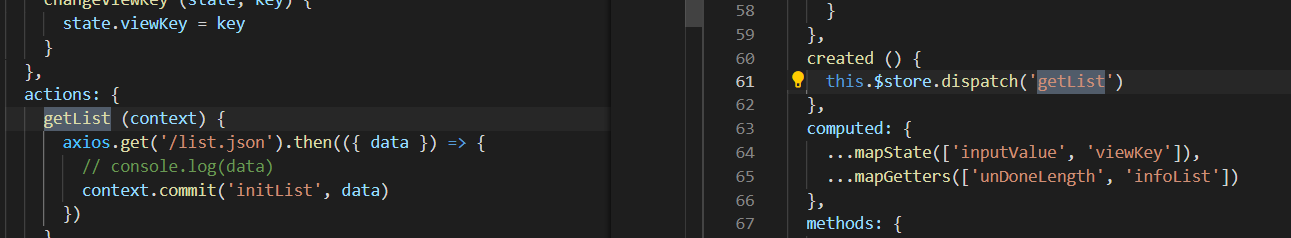
（1） 通过 vue ui 命令打开可视化面板，创建新项目 vuex-demo2

（2）安装 vuex 依赖包 npm install vuex axios ant-design-vue –S

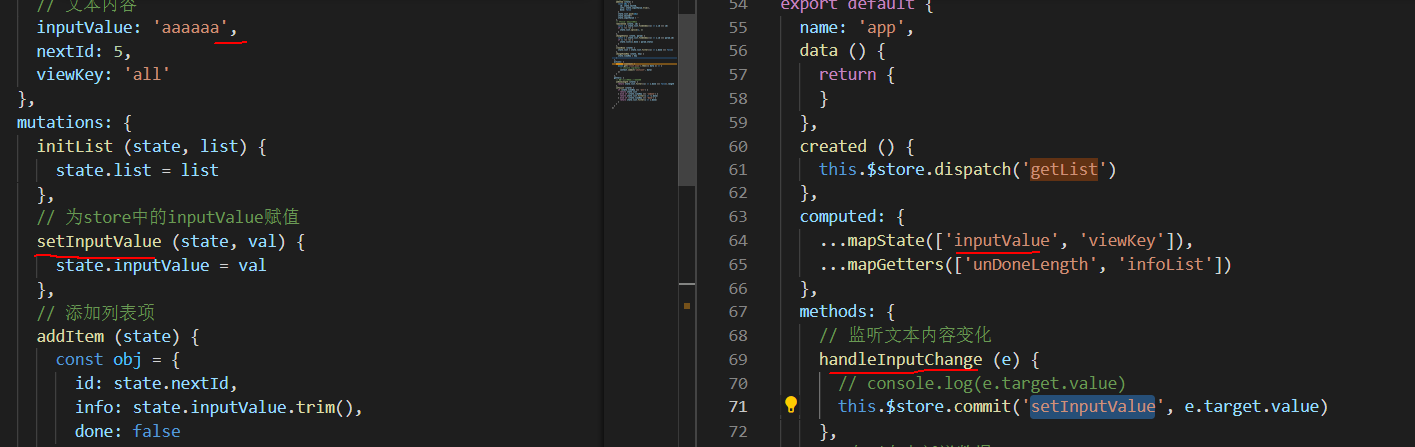
（3） 实现 Todos 基本布局（基于已有样式模板）

2. 完成具体功能

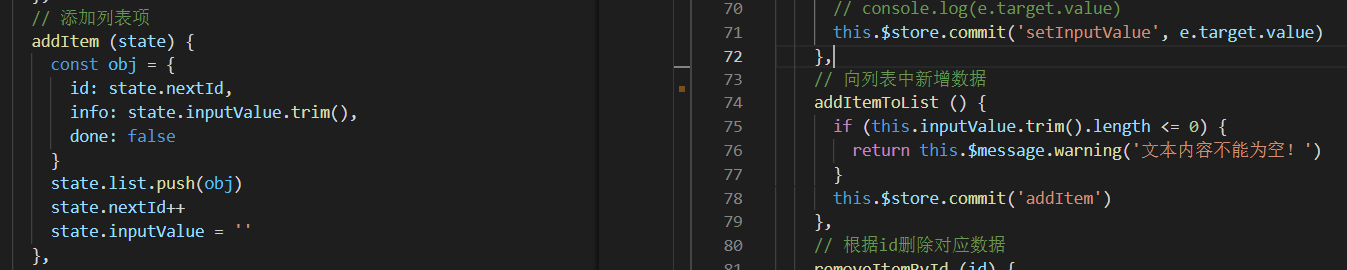
①动态加载任务列表数据



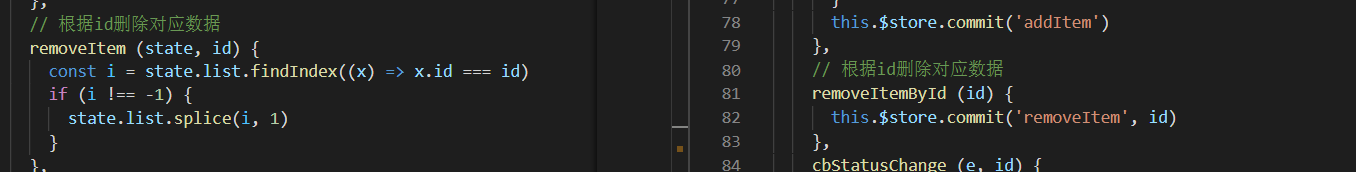
②实现文本框与store数据的双向同步



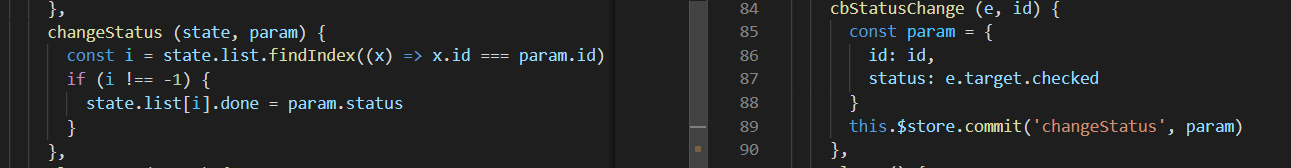
③完成添加任务事项的操作



④完成删除任务事项的操作

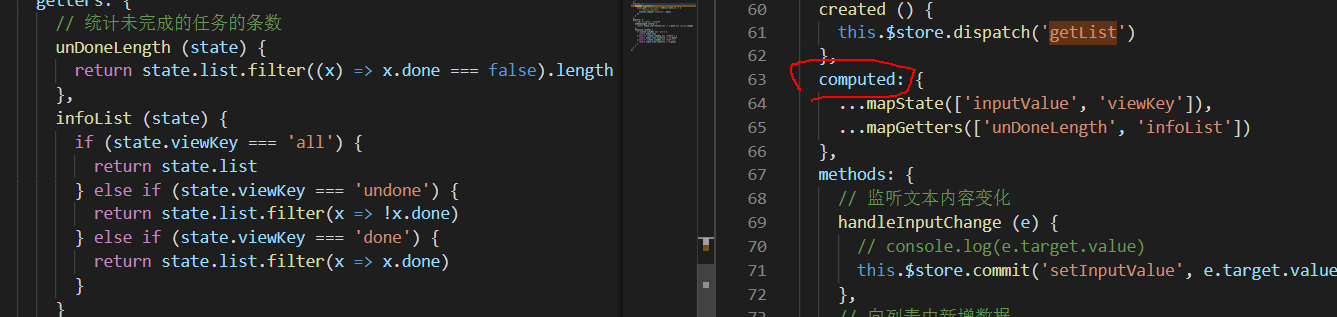


⑤动态绑定复选框的选中状态

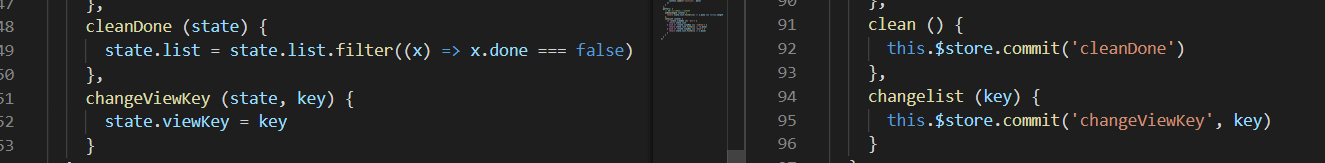


⑥修改任务事项的完成状态

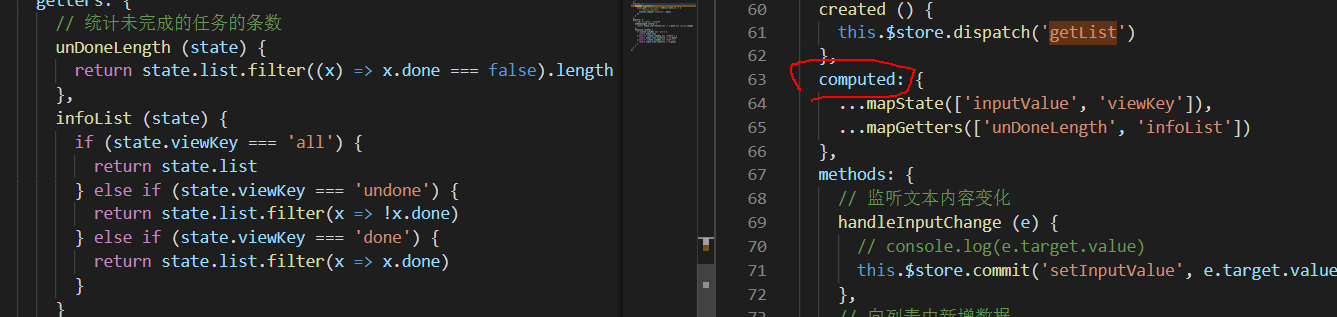
⑦统计未完成的任务的条数



⑧清除已完成的任务事项



⑨实现任务列表数据的动态切换



实现：





