1. 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。
2. 不能直接改变 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，从而让我们能够实现一些工具帮助我们更好地了解我们的应用。
3. Vuex 使用 单一状态树、，用一个对象就包含了全部的应用层级状态，作为一个『唯一数据源(SSOT)』而存在。
4. Vuex 通过 store 选项，提供了一种机制将状态从根组件『注入』到每一个子组件中（需调用 Vue.use(Vuex)）：  
   const app = new Vue({  
    el: '#app',  
    // 把 store 对象提供给 “store” 选项，这可以把 store 的实例注入所有的子组件  
    store,  
    template:  
    <div class="app">  
    <counter></counter>  
    </div>  
   })
5. mapState 辅助函数：帮助我们快速生成计算属性。
6. mapGetters 辅助函数仅仅是将 store 中的 getters 映射到局部计算属性
7. Vuex 中的 mutations 非常类似于事件：每个 mutation 都有一个字符串的 事件类型 (type) 和 一个 回调函数 (handler)
8. mutation 必须是同步函数，使用常量替代 Mutation 事件类型
9. 在组件中可使用 this.$store.commit('xxx') 提交 mutation，或者使用 mapMutations 辅助函数将组件中的 methods 映射为 store.commit 调用（需要在根节点注入 store）
10. Action 提交的是 mutation，而不是直接变更状态。Action 可以包含任意异步操作。
11. Action 通过 store.dispatch 方法触发：store.dispatch(‘increment')
12. 在组件中使用 this.$store.dispatch('xxx') 分发 action，或者使用 mapActions 辅助函数将组件的 methods 映射为 store.dispatch 调用（需要先在根节点注入 store）
13. mutation和Action最大的区别：前者必须为同步函数，后者可以为异步操作。
14. Vuex 允许我们将 store 分割到模块（module）。每个模块拥有自己的 state、mutation、action、getters、甚至是嵌套子模块——从上至下进行类似的分割  
    const moduleA = {  
     state: { ... },  
     mutations: { ... },  
     actions: { ... },  
     getters: { ... }  
    }  
      
    const moduleB = {  
     state: { ... },  
     mutations: { ... },  
     actions: { ... }  
    }  
      
    const store = new Vuex.Store({  
     modules: {  
     a: moduleA,  
     b: moduleB  
     }  
    })  
      
    store.state.a // -> moduleA 的状态  
    store.state.b // -> moduleB 的状态