Vuex

<https://segmentfault.com/a/1190000009404727>

<https://segmentfault.com/a/1190000012015742>

<https://vuex.vuejs.org/zh/guide/>

1.



2.action:



3.不懂：

https://vuex.vuejs.org/zh/guide/actions.html#%E5%9C%A8%E7%BB%84%E4%BB%B6%E4%B8%AD%E5%88%86%E5%8F%91-action



5.不懂：



6. 单状态树和模块化并不冲突——在后面的章节里我们会讨论如何将状态和状态变更事件分布到各个子模块中

7. 每当 store.state.count 变化的时候, 都会重新求取计算属性，并且触发更新相关联的 DOM。

8. 通过在根实例中注册 store 选项，该 store 实例会注入到根组件下的所有子组件中，且子组件能通过 this.$store 访问到

9. mapState 辅助函数

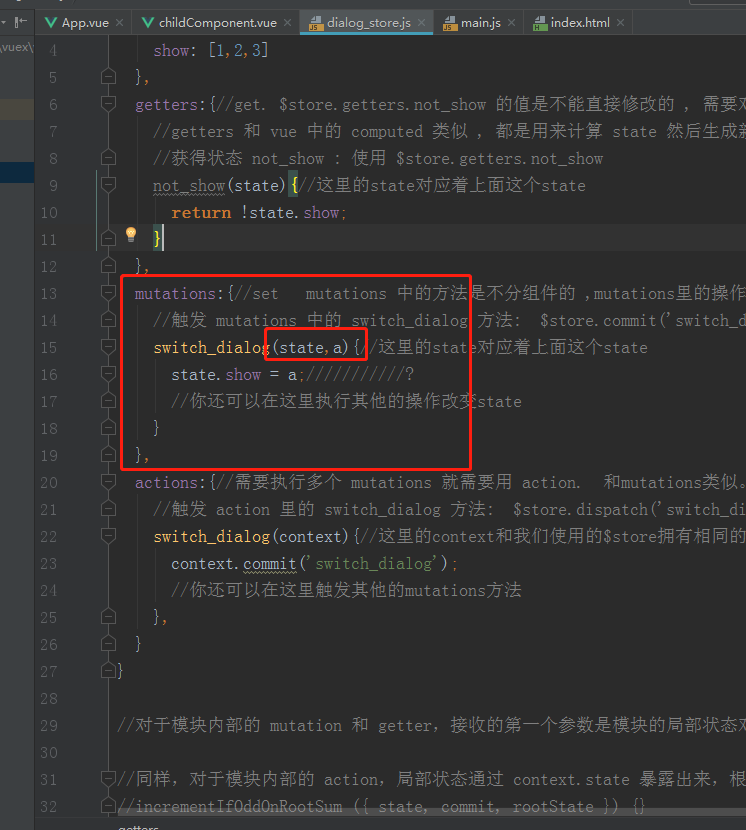
// 为了能够使用 `this` 获取局部状态，必须使用常规函数

10. Vuex Store是响应式的，当Vue组件从store中读取状态（state选项）时，若store中的状态发生更新时，他会及时的响应给其他的组件（类似双向数据绑定） 而且不能直接改变store的状态，改变状态的唯一方法就是，显式地提交更改（mutations选项）

11. 把getters看成是store的计算属性

12.要传参的话这里要加第二个参数：

<https://www.cnblogs.com/first-time/p/6815036.html>



13.action里如何嵌套触发action?

14.总结：

　　　　各个类型的 API各司其职，mutation 只管存，你给我（dispatch）我就存；action只管中间处理，处理完我就给你，你怎么存我不管；Getter 我只管取，我不改的。　action放在了 methods 里面，说明我们应该把它当成函数来用（讲道理，钩子函数也应该可以的） mutation是写在store里面的，这说明，它就是个半成品，中间量，我们不应该在外面去操作它。getter写在了 computed 里面，这说明虽然 getter我们写的是函数，但是我们应该把它当成计算属性来用。

15. 由于 store 中的状态是响应式的，在组件中调用 store 中的状态简单到仅需要在计算属性中返回即可。触发变化也仅仅是在组件的 methods 中提交 mutation。

16. 每个应用将仅仅包含一个 store 实例

17. 从 store 实例中读取状态最简单的方法就是在计算属性中返回某个状态：

18. 通过在根实例中注册 store 选项，该 store 实例会注入到根组件下的所有子组件中，且子组件能通过 this.$store 访问到。

19. 使用 Vuex 并不意味着你需要将所有的状态放入 Vuex, 如果有些状态严格属于单个组件，最好还是作为组件的局部状态。你应该根据你的应用开发需要进行权衡和确定。

20. 就像计算属性一样，getter 的返回值会根据它的依赖被缓存起来，且只有当它的依赖值发生了改变才会被重新计算。

21. 注意，getter 在通过属性访问时是作为 Vue 的响应式系统的一部分缓存其中的

注意，getter 在通过方法访问时，每次都会去进行调用，而不会缓存结果。

mapGetters 辅助函数仅仅是将 store 中的 getter 映射到局部计算属性：

22. mutation

提交载荷（Payload）//即向函数里传递要传递的数据

你可以向 store.commit 传入额外的参数，即 mutation 的 载荷（payload）：在大多数情况下，载荷应该是一个对象，这样可以包含多个字段并且记录的 mutation 会更易读：

既然 Vuex 的 store 中的状态是响应式的，那么当我们变更状态时，监视状态的 Vue 组件也会自动更新。这也意味着 Vuex 中的 mutation 也需要与使用 Vue 一样遵守一些注意事项：

最好提前在你的 store 中初始化好所有所需属性。

当需要在对象上添加新属性时，你应该

使用 Vue.set(obj, 'newProp', 123), 或者

以新对象替换老对象。例如，利用 stage-3 的对象展开运算符我们可以这样写：

一条重要的原则就是要记住 mutation 必须是同步函数

23.action

Action 函数接受一个与 store 实例具有相同方法和属性的 context 对象, 当我们在之后介绍到 Modules 时，你就知道 context 对象为什么不是 store 实例本身了。

Actions 支持同样的载荷方式和对象方式进行分发//即能够传参进去

首先，你需要明白 store.dispatch 可以处理被触发的 action 的处理函数返回的 Promise，并且 store.dispatch 仍旧返回 Promise：

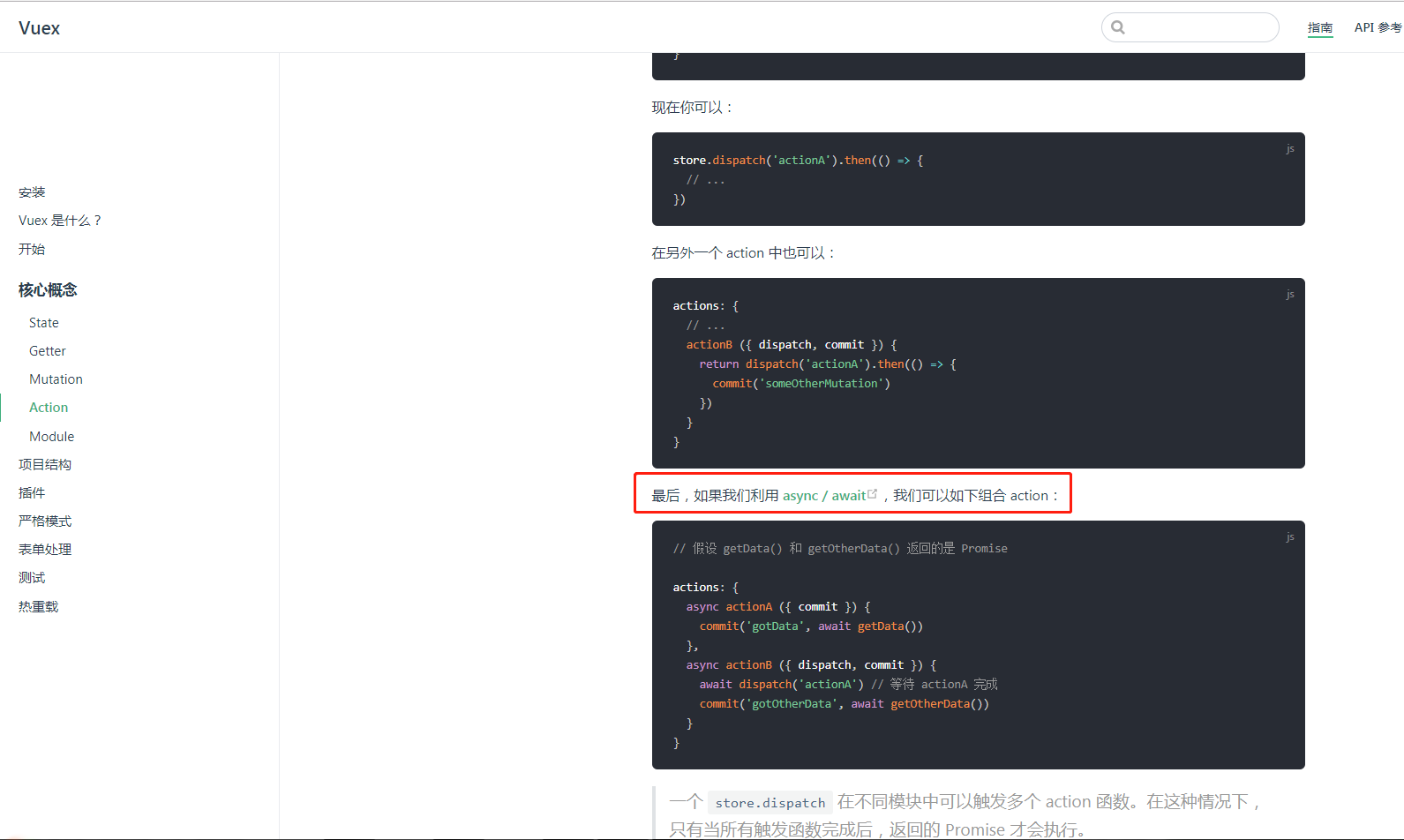
一个 store.dispatch 在不同模块中可以触发多个 action 函数。在这种情况下，只有当所有触发函数完成后，返回的 Promise 才会执行。

不懂这部分和Promise相关的内容

async [ə'zɪŋk] 相关的内容：

https://segmentfault.com/a/1190000013292562?utm\_source=channel-newest





15. Vuex 允许我们将 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：

嵌套模块继承父模块的命名空间

启用了命名空间的 getter 和 action 会收到局部化的 getter，dispatch 和 commit。（即启用了命名空间后getter ，dispatch 和 commit被局部化了，若要访问根模板的内容要将其作为参数传进去；若需要在全局命名空间内分发 action 或提交 mutation，将 { root: true } 作为第三参数传给 dispatch 或 commit 即可） 换言之，你在使用模块内容（module assets）时不需要在同一模块内额外添加空间名前缀。更改 namespaced 属性后不需要修改模块内的代码

如果你希望使用全局 state 和 getter，rootState 和 rootGetter 会作为第三和第四参数传入 getter，也会通过 context 对象的属性传入 action。

模块动态注册: 模块动态注册功能使得其他 Vue 插件可以通过在 store 中附加新模块的方式来使用 Vuex 管理状态。例如，vuex-router-sync 插件就是通过动态注册模块将 vue-router 和 vuex 结合在一起，实现应用的路由状态管理。

模块重用

有时我们可能需要创建一个模块的多个实例，例如：

创建多个 store，他们公用同一个模块 (例如当 runInNewContext 选项是 false 或 'once' 时，为了在服务端渲染中避免有状态的单例)

在一个 store 中多次注册同一个模块

如果我们使用一个纯对象来声明模块的状态，那么这个状态对象会通过引用被共享，导致状态对象被修改时 store 或模块间数据互相污染的问题。

实际上这和 Vue 组件内的 data 是同样的问题。因此解决办法也是相同的——使用一个函数来声明模块状态（仅 2.3.0+ 支持）：

19.插件

有时候插件需要获得状态的“快照”，比较改变的前后状态。想要实现这项功能，你需要对状态对象进行深拷贝：

生成状态快照的插件应该只在开发阶段使用，使用 webpack 或 Browserify，让构建工具帮我们处理：

Vuex 自带一个日志插件用于一般的调试: 日志插件还可以直接通过 <script> 标签引入，它会提供全局方法 createVuexLogger。

要注意，logger 插件会生成状态快照，所以仅在开发环境使用。

20.严格模式

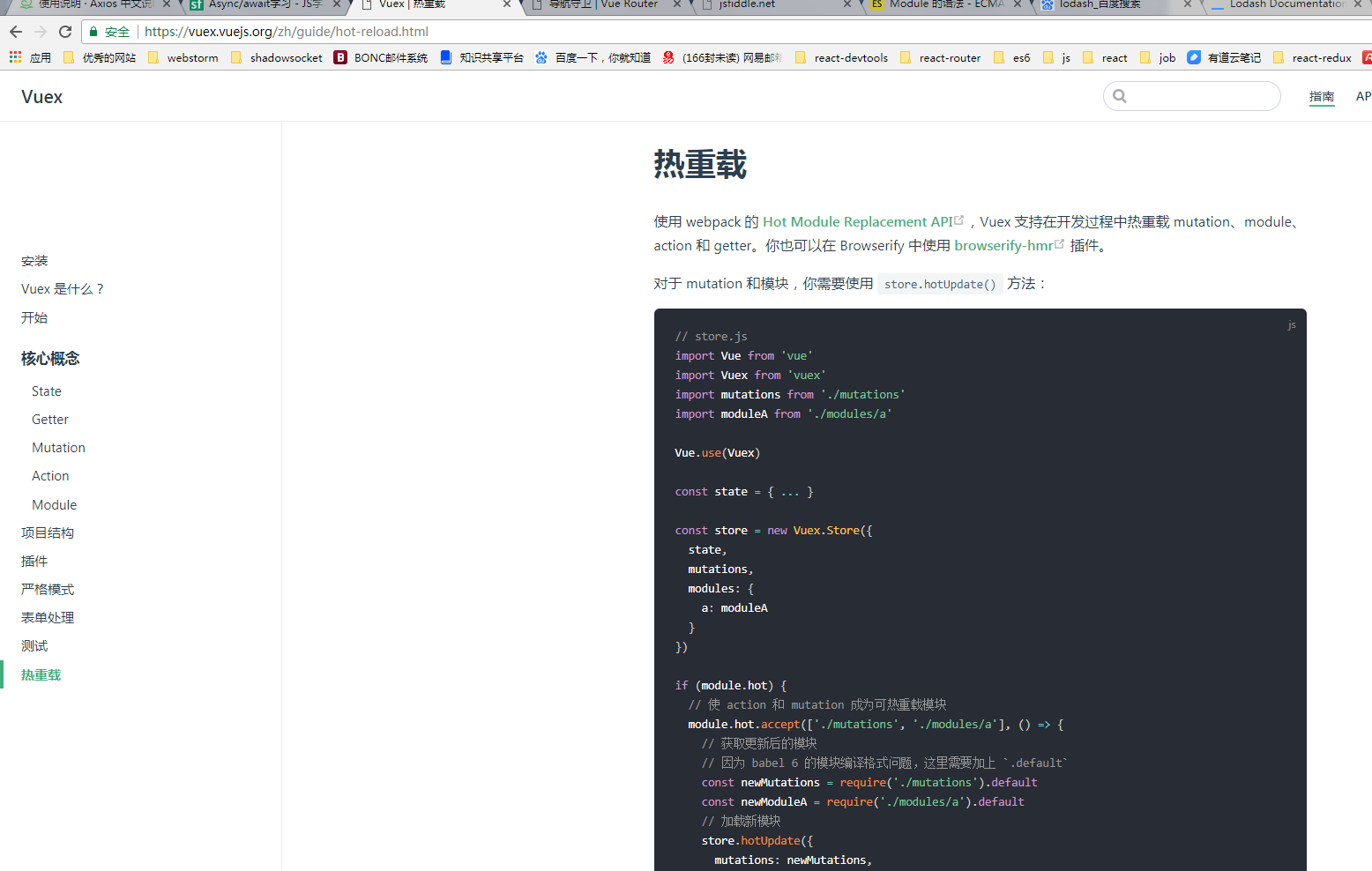
不要在发布环境下启用严格模式

21.表单处理

当在严格模式中使用 Vuex 时，在属于 Vuex 的 state 上使用 v-model 会比较棘手，用“Vuex 的思维”去解决这个问题的方法是：给 <input> 中绑定 value，然后侦听 input 或者 change 事件，在事件回调中调用 action:

必须承认，这样做比简单地使用“v-model + 局部状态”要啰嗦得多，并且也损失了一些 v-model 中很有用的特性。另一个方法是使用带有 setter 的双向绑定计算属性：

22.热重载 –不懂



23. <https://segmentfault.com/a/1190000013292562?utm_source=channel-newest>