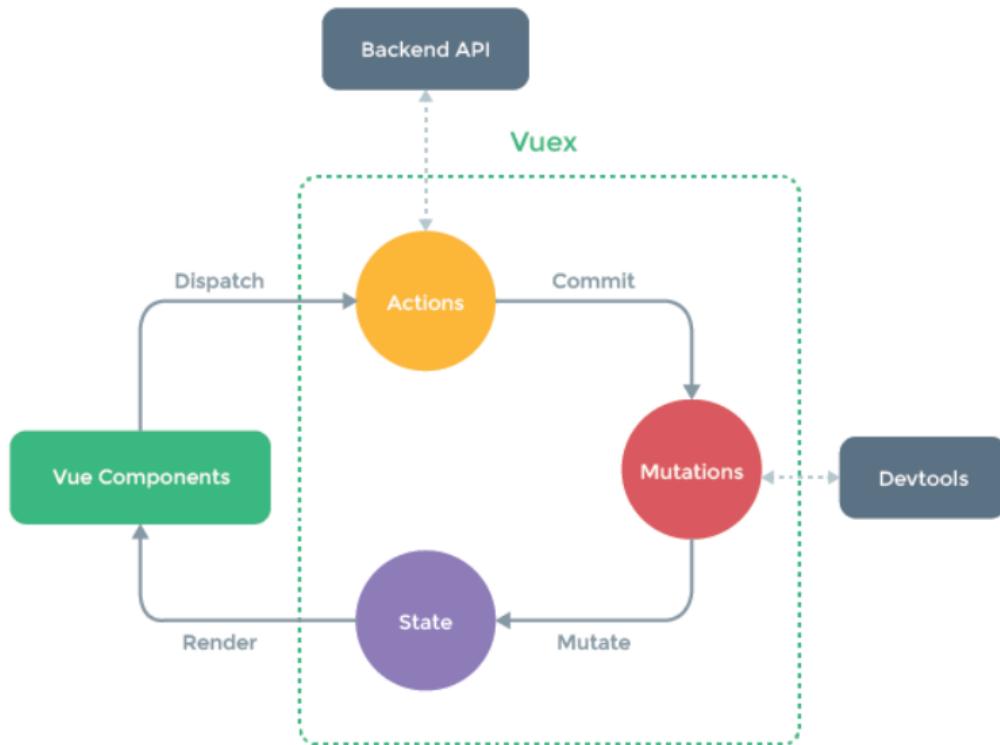


一. 状态管理 Vuex

1. Vuex使用

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。



(1)state:单一状态树 ,每个应用将仅仅包含一个 store 实例。

*this.\$store.state.状态名字

*...mapState(["title"])

(2)getters:可以从store 中的 state 中派生出一些状态， getters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。

*可以认为是 store 的计算属性

*this.\$store.getters.计算属性名字

*...mapGetters(["getFilms"])

(3)mutations: 更改 Vuex 的 store 中的状态的唯一方法是提交 mutation。

*常量的设计风格

```
[SOME_MUTATION] (state) {  
  // mutate state  
}
```

*必须是同步函数

```
*this.$store.commit("type","payload");
```

(4)actions:

*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。

```
*this.$store.dispatch("type","payload")
```

(5)

```
1  const store = new Vuex.Store({  
2    state: {  
3      count: 0  
4    },  
5    mutations: {  
6      increment (state ,payload) {  
7  
8      }  
9    },  
10   actions: {  
11     increment (context, payload) {  
12       context.commit('increment')  
13     }  
14   }  
15 })
```

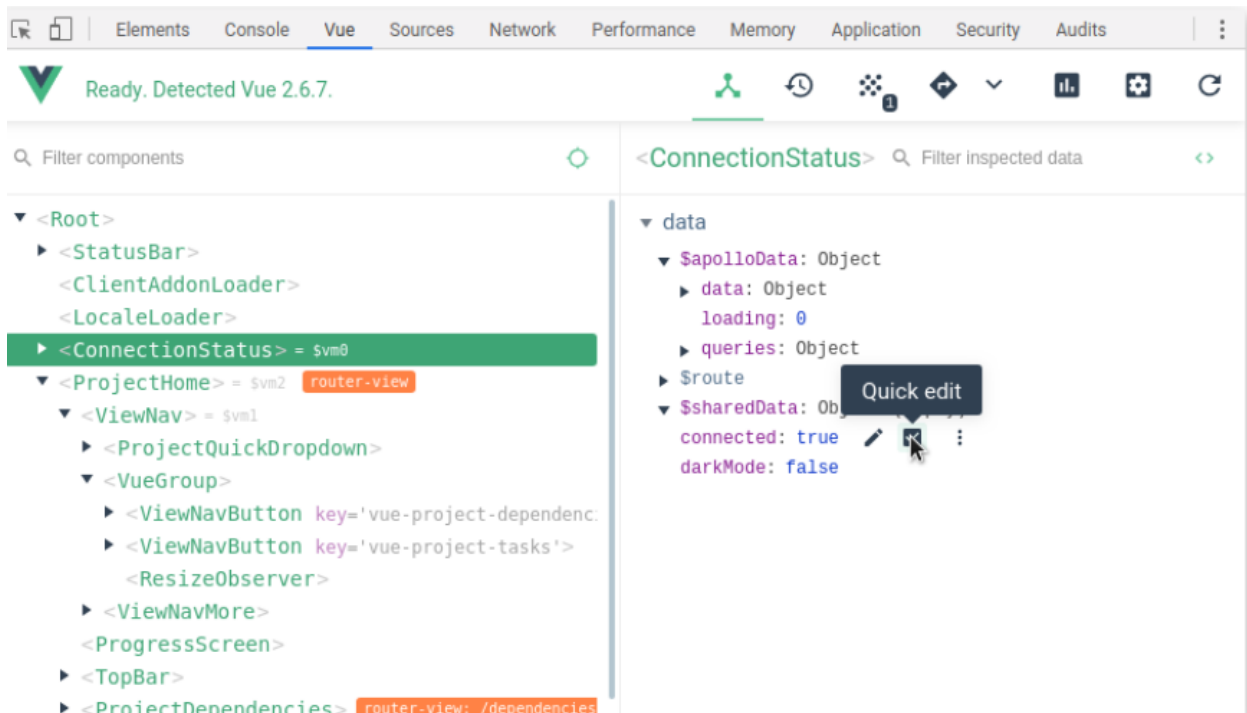
2. 注意:

(1)应用层级的状态应该集中到单个 store 对象中。

(2)提交 mutation 是更改状态的唯一方法，并且这个过程是同步的。

(3)异步逻辑都应该封装到 action 里面。

3. vue chrome devtools



二. vuex在项目中的使用

1. 复杂非父子通信
2. 异步数据快照

三. vuex持久化

<https://github.com/robinvdvleuten/vuex-persistedstate>

```
import createPersistedState from "vuex-persistedstate";

const store = new Vuex.Store({
  ...
  plugins: [createPersistedState({
    storage: window.sessionStorage, //默认是localStorage
    reducer(val) {
      return {
        // 只储存state中的user
        user: val.user
      }
      //默认是全部存储
    }
  })]
})
```

