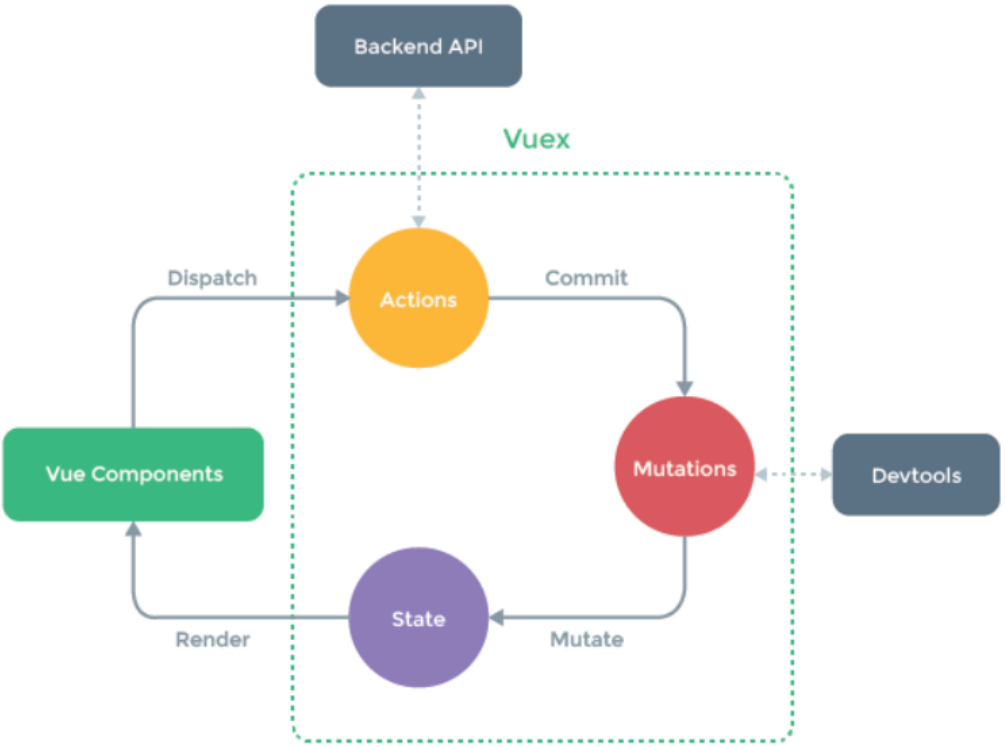
一. 状态管理 Vuex

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)

```
const store = new Vuex.Store({
state: {
count: 0
},
mutations: {
increment (state ,payload) {

}
},
 actions: {
 increment (context, payload) {
 context.commit('increment')
 }
 }
 })
```

(6) 模块分割

```
var moduleA = {
  namespaced:true,
  state :{
    isShow:true
  },
  mutations: {
    show(state){
      state.isShow = true
    },
    hide(state){
      state.isShow = false;
    }
  },
}
```

```
modules: {
  moduleA
}
```

```
...mapState('moduleA',["isShow"])
```

```
this.$store.commit("moduleA/show")
```

如果希望你的模块具有更高的封装度和复用性，你可以通过添加 `namespaced: true` 的方式使其成为带命名空间的模块。当模块被注册后，它的所有 getter、action 及 mutation 都会自动根据模块注册的路径调整命名。例如：

```js
const store = new Vuex.Store({
  modules: {
    account: {
      namespaced: true,

      // 模块内容（module assets）
      state: { ... }, // 模块内的状态已经是嵌套的了，使用 `namespaced` 属性不会对其产生影响
      getters: {
        isAdmin () { ... } // -> getters['account/isAdmin']
      },
      actions: {
        login () { ... } // -> dispatch('account/login')
      },
      mutations: {
        login () { ... } // -> commit('account/login')
      },
```

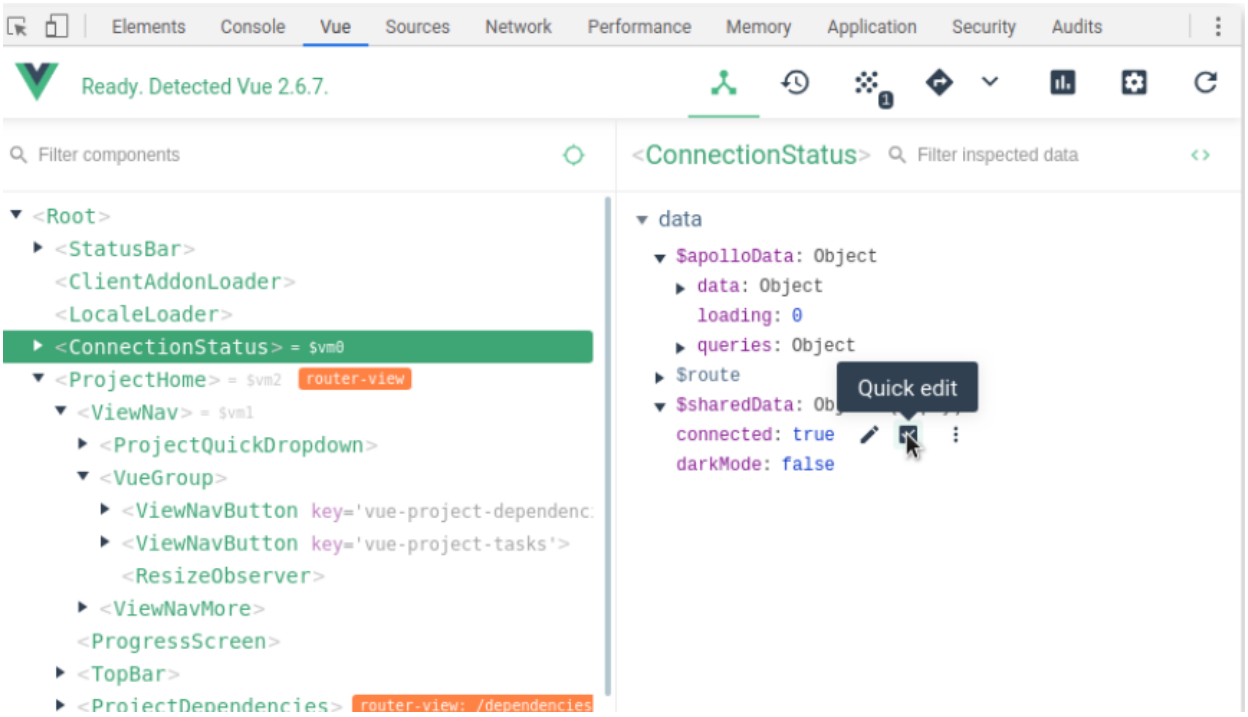
2. 注意:

(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
      }
  })]
})
```