执行初始化方法 install

function install(\_Vue) {

    Vue = \_Vue;

    // 混入store

    Vue.mixin({

        beforeCreate() {

            if (this.$options.store) {

                Vue.prototype.$store = this.$options.store;

            }

        }

    })

}

// 2.实现Store

class Store {

    constructor(options = {}) {

        // 响应化处理

        this.state = new Vue({

            data: options.state

        })

        this.mutations = options.mutations || {};

        this.actions = options.actions || {};

    }

    // type是mutations中的函数名

    commit = (type, arg) => {

        this.mutations[type](this.state, arg)

    }

    dispatch(type, arg) {

        this.actions[type]({

            commit: this.commit,

            state: this.state

        }, arg)

    }

}

新建store类 主要编写业务逻辑

export default new Vuex.Store({

    state: { count: 10 },

    mutations: {

      increment(state) {

        state.count += 1;

      }

    },

    actions: {

        asyncAdd({commit}){

            commit("increment");

        },

        // ctx: {state,commit,dispatch,getters}

      increment({ getters, commit }) {

        // 添加业务逻辑

        if (getters.left > 0) {

          commit("increment");

          return true;// 返回结果

        }

        return false;// 返回结果

      },

      asyncIncrement({ dispatch }) {

        // 异步逻辑返回Promise

        return new Promise(resolve => {

          setTimeout(() => {

            // 复用其他action

            resolve(dispatch("increment"));

          }, 1000);

        });

      },

    }

  })