vue-x的理解

vue-x的理解

使用vue-x的步骤

安装

import引入

use注册到Vue

注入vue实例

使用数据

state

getters

修改数据

mutations——commit

actions——dispatch

vue-x的理解

vue-x是数据仓库，数据影响组件，所以vue-x还是视图状态管理。

使用vue-x的步骤

安装>**npm i vuex**

import引入

use注册到Vue

注入vue实例，组件中就有个$store属性



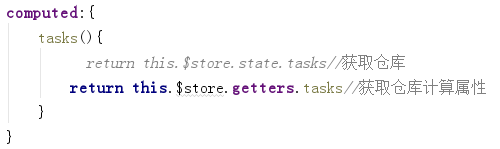
获取数据——state:{}

原始数据

注意：数据不要放在data里面，因为data在组件的时候执行一次，数据更新，不执。，放在watch或computed下面，最好放到computed下面。

获取数据——getters（state）{return }

计算后数据，依赖于state原始数据



修改数据——同步——mutations(state,数据)——commit

过程：

通知仓库：this.$store.commit('remove', id);

仓库修改数据：调用mutations里面的方法remove

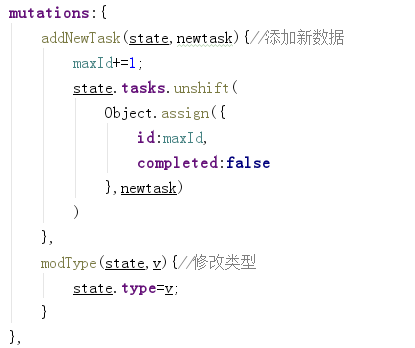
参数：

第一个参数：就是当前仓库的state
第二个参数：提交任务的时候，传过来的数据

返回值：没有返回值

原理：commit通知仓库修改数据 => store就会调用mutations下的remove方法

给仓库数据：



修改数据——异步——actions(store,数据)——dispatch

处理异步任务，和mutation不同之处，在于可以返回值

过程：

通知仓库：this.$store.dispatch('remove', id);

仓库操作异步：调用actions里面的方法remove

参数：

第一个参数：就是当前仓库的store
第二个参数：提交任务的时候，传过来的数据

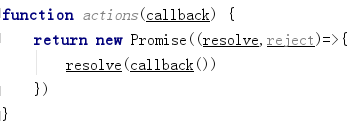
将异步操作结果通知给mutations。this.$store.commit('remove', id);

返回值操作：有返回值，返回值是Promise对象——组件根据返回值做进一步操作

原理：dispatch通知仓库操作异步 => store就会调用actions下的remove方法，commit通知仓库修改数据 => 组件接收actions的返回值，做进一步操作

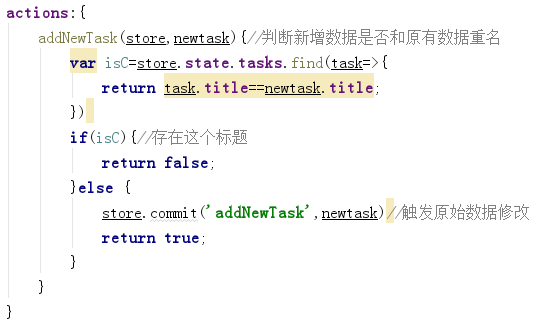
actions的理解：

包装数据。返回promise对象。



给参数数据：

如果存在这个数据重名的元素，就不添加



返回结果给组件：



mutations 和 actions的区别

mutations处理数据，（可以操作异步操作，但不返回值）

actions处理异步操作，通知mutations处理数据，并且返回值

额外知识：

图片地址的问题：组件里面的地址不能被读取

因为webpack不会去解析图片地址。可以图片放到static，浏览器解析，给服务器执行，发请求，映射到static下面。

所以一个文件放在哪里，是要去看是被浏览器解析，还是被webpack解析。