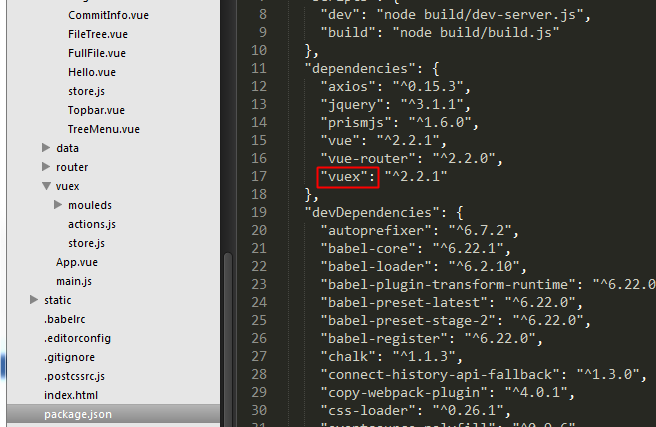
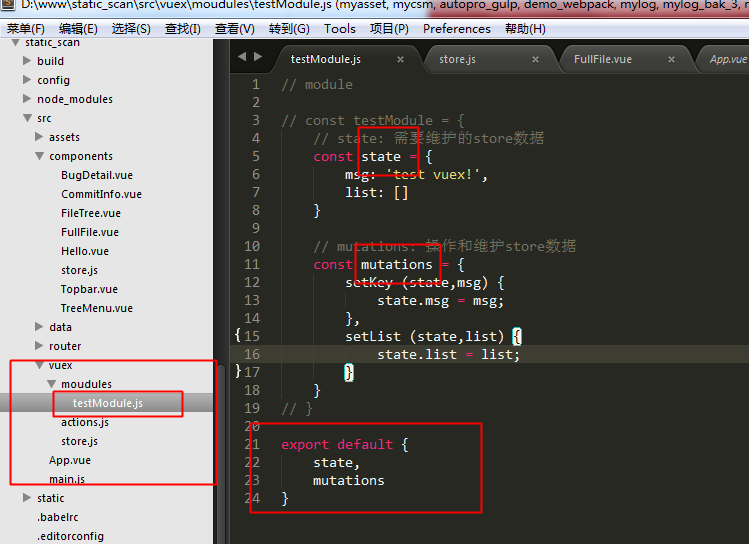
## 开始

下载包

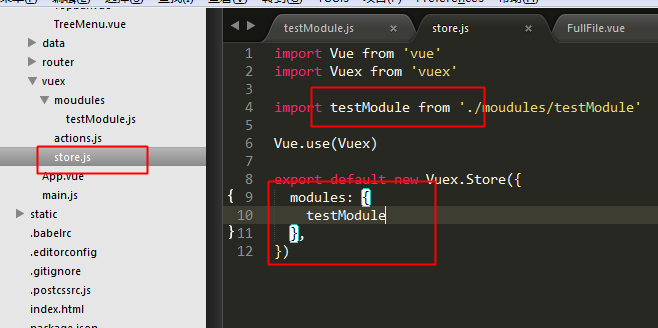


### 编写vuex组件（Vuex项目结构）

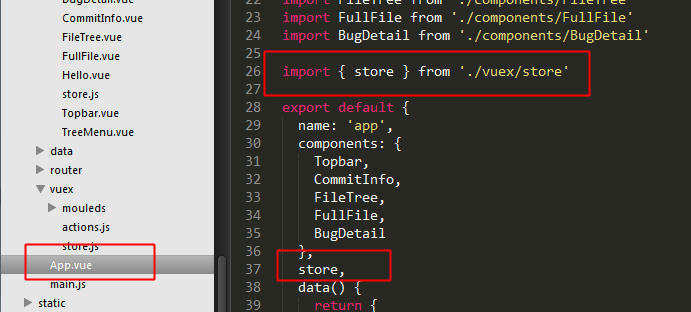
定义Module



store.js核心

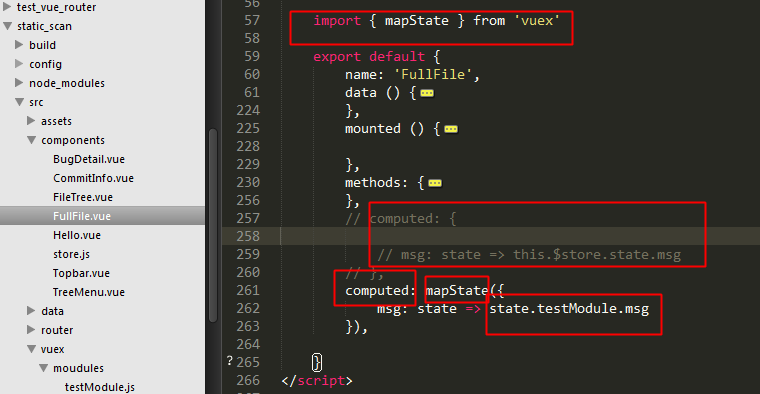


入口文件引入store



### 使用vuex管理状态





## 核心概念

#### State：store状态，数据管理

Vuex 使用 **单一状态树** —— 是的，用一个对象就包含了全部的应用层级状态。至此它便作为一个『唯一数据源([SSOT](https://en.wikipedia.org/wiki/Single_source_of_truth))』而存在。这也意味着，每个应用将仅仅包含一个 store 实例。

#### Getters：从store中的state中派生出的状态

getters: {

doneTodos: state => {

return state.todos.filter(todo => todo.done)

}

}

Getters 会暴露为 store.getters 对象：

store.getters.doneTodos // -> [{ id: 1, text: '...', done: true }]

Getters 也可以接受其他 getters 作为第二个参数：

getters: {

// ...

doneTodosCount: (state, getters) => {

return getters.doneTodos.length

}

}

#### Mutations：更改 Vuex 的 store 中的状态的唯一方法是提交 mutation

1. 接收state参数
2. this.$store.commit('increment')
3. 你可以向 store.commit 传入额外的参数，即 mutation 的 **载荷（payload）**：store.commit('increment', obj)
4. 一条重要的原则就是要记住 **mutation 必须是同步函数**
5. 在 mutation 中混合异步调用会导致你的程序很难调试。例如，当你能调用了两个包含异步回调的 mutation 来改变状态，你怎么知道什么时候回调和哪个先回调呢

#### Actions：类似于 mutation，不同在于：

* 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。

Action 函数接受一个与 store 实例具有相同方法和属性的 context 对象，因此你可以调用 context.commit 提交一个 mutation，或者通过 context.state 和 context.getters 来获取 state 和 getters。

mutations: {

increment (state) {

state.count++

}

},

actions: {

increment (context) {

context.commit('increment')

}

}

我们会经常会用到 ES2015 的 [参数解构](https://github.com/lukehoban/es6features#destructuring) 来简化代码（特别是我们需要调用 commit 很多次的时候）：

actions: {

increment ({ commit }) {

commit('increment')

}

}

Action 通过 store.dispatch 方法触发：

store.dispatch('increment')

#### Modules:

使用单一状态树，导致应用的所有状态集中到一个很大的对象。但是，当应用变得很大时，store 对象会变得臃肿不堪。

为了解决以上问题，Vuex 允许我们将 store 分割到**模块（module）**。每个模块拥有自己的 state、mutation、action、getters、甚至是嵌套子模块——从上至下进行类似的分割：

对于模块内部的 action，context.state 是局部状态，根节点的状态是 context.rootState

对于模块内部的 getter，根节点状态会作为第三个参数

const moduleA = {

// ...

getters: {

sumWithRootCount (state, getters, rootState) {

return state.count + rootState.count

}

}

}