VUEX教程

针对cli脚手架搭建的项目；

必需流程：

1 安装vue（参考官网）

# 最新稳定版  
$ npm install vue

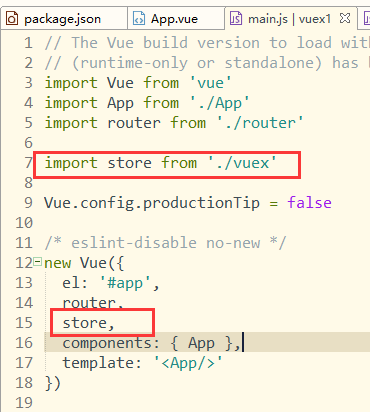
# 全局安装 vue-cli  
$ npm install --global vue-cli  
# 创建一个基于 webpack 模板的新项目  
$ vue init webpack my-project  
# 安装依赖，走你  
$ cd my-project  
$ npm run dev

2 安装vuex

$ npm install vuex –save

3 项目中新建vuex文件夹，存放相关代码，并用index.js导出stroe

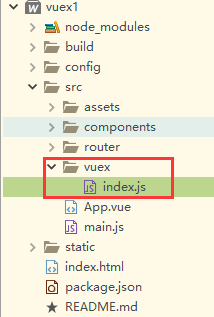
4 项目入口文件（mian.js）中引用并注册stroe



# 基础用法

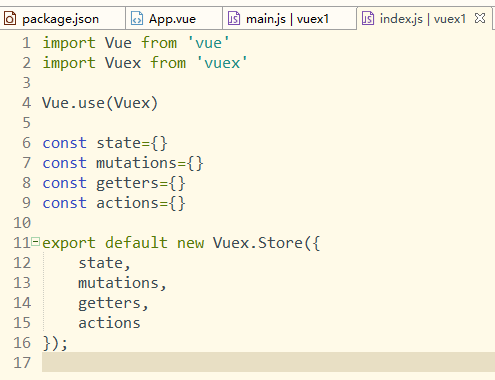
——所有store代码都在一个js文件中

## 目录结构：



只有一个index.js存放所有代码

## Index.js文件结构



## 文件说明

### state

格式：json格式；

作用：store 的状态树；

本人理解为中存放数据的地方。

#### 定义state

例：const state={

nub:1000,

name:'Lee'

}

#### 组件中使用state

##### 使用方法一

template中直接写入----> {{$store.state}}

##### 使用方法二

computed中映射出mapState，如下图：共3步



### mutations

格式：json格式；

作用：其中的function是更改state的唯一方法;

可否异步：否

#### 定义mutations

例：const mutations={

JIA(state,n){

state.nub+=n;

},

JIAN(state,n){

state.nub-=n;

}

}

参数说明：

Mutations中的方法中第一个参数，为自带的默认参数state，这个state为上文中const的state，即store中的数据；为必传参数，不可省略。

Mutations中的方法中，可传入第二个参数，官方叫做提交载荷（Payload）；这个参数的值为调用时传入的；格式一般为json，如；payload ={ n : 1, name : ‘Lee’ }

const mutations={

JIA(state, payload){

state.nub+= payload;

}

JIAN(state, payload){

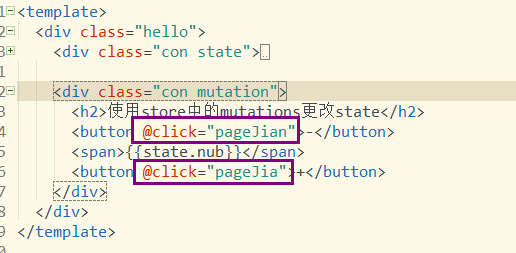
state.nub+= payload.n;

}

}

#### 组件中调用mutations

1 先在组件中定义 点击事件



2 在事件中调用mutation

##### 调用方法一

不映射mutation中的function 直接调用

this.$store.commit（‘方法名’,‘参数’）



##### 调用方法二

1 先引入mutations

2 映射出所需方法

3 事件中调用



### actions

格式：json格式；

作用：由于**mutation 必须同步执行，为突破此限制，action可进行异步操作，提交mutation**;

可否异步：可

#### 定义actions

const actions={

JIA\_ACT({commit},payload){

commit('JIA',payload)

},

JIAN\_ACT({commit},payload){

commit('JIAN',payload)

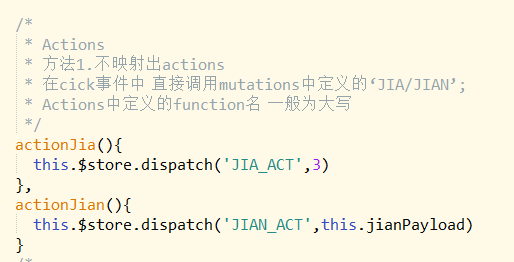
}

}

##### 调用方法一

不映射action中的function 直接调用

this.$store.dispatch('方法名','参数')

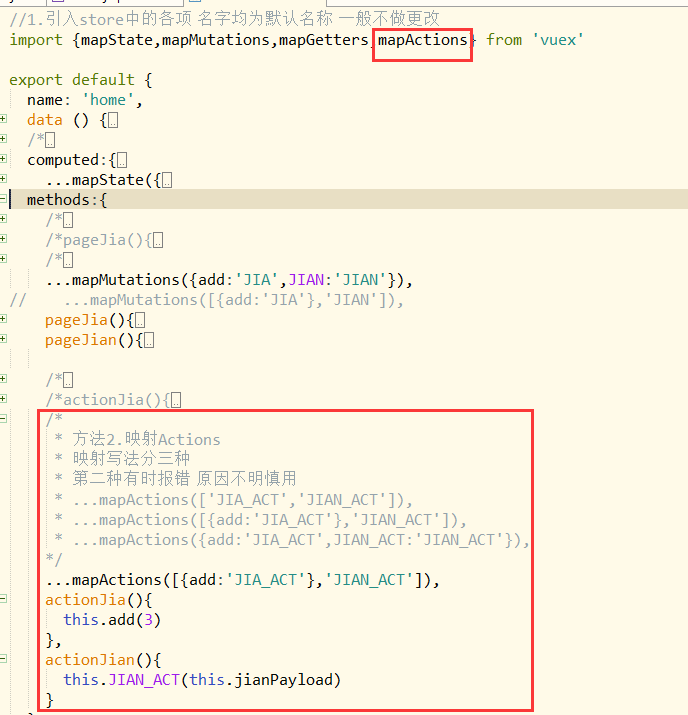


##### 调用方法二

1 先引入actions

2 映射出所需方法

3 事件中调用



### getters

格式：json格式；

作用：本人理解为类似过滤器或computed;不对数据进行更改，返回一个新属性

#### 定义getter

const getters={

STATE\_GET(state){

return `====get====${JSON.stringify(state)}`

},

NUB\_GET(state){

return `====get====${state.nub}`

},

INFO\_GET(state){

return `====get====${state.info}`

}

}

#### 组件中使用getter

##### 使用方法一

template中直接写入----> {{$store.getters.STATE\_GET}}

##### 使用方法二

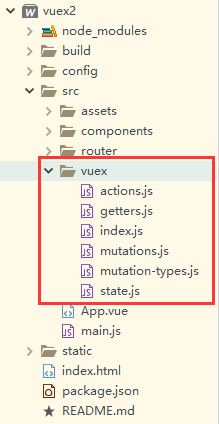
computed中映射出mapGetters，如下图：共3步



# 拆分文件

——把index.js中的代码拆分成4个文件，并添加mutations-type.js

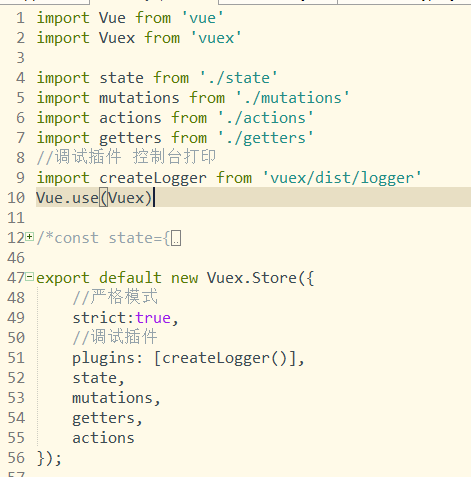
## 目录结构：



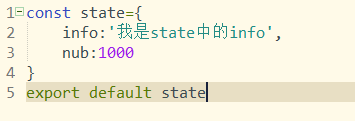
## Index.js文件结构

1 导入拆分出去的文件

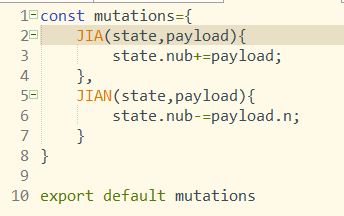
2 最后 new Vuex.Store 并导出



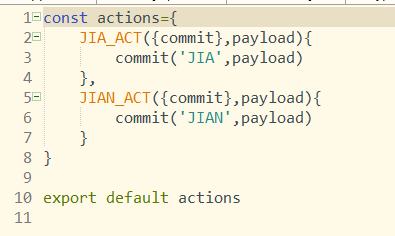
### state



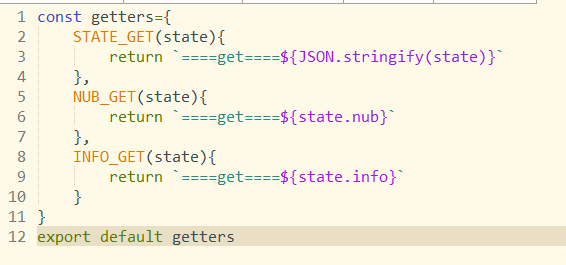
### mutations



### actions



### getters

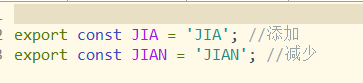


### types

作用：对整个 app 包含的 mutation 一目了然，方便多人共同开发

一般单独放到一个js文件中保存

##### 文件格式：



##### 使用方法

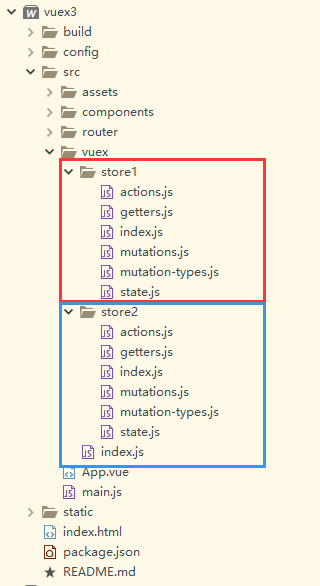
1 组件中引入type文件

2 利用type映射mutations



# 拆分模块

## 目录结构：



## Index.js文件结构

1 导入模块1---- store1、 模块2---- store2

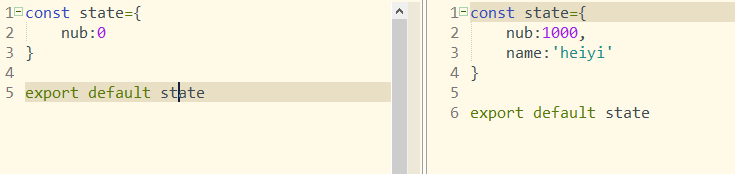
2 最后 new Vuex.Store 并导出



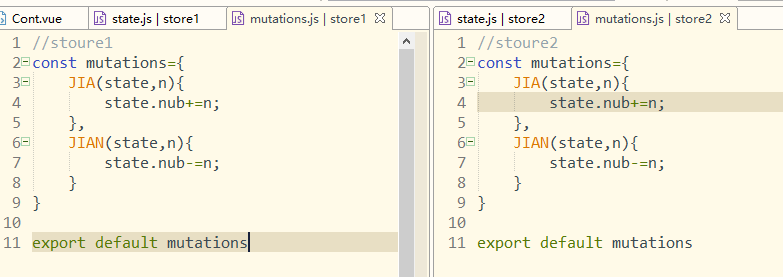
## 文件内容：

此demo中stroe1与store2中的state等文件中的变量名称故意设置的完全一样。

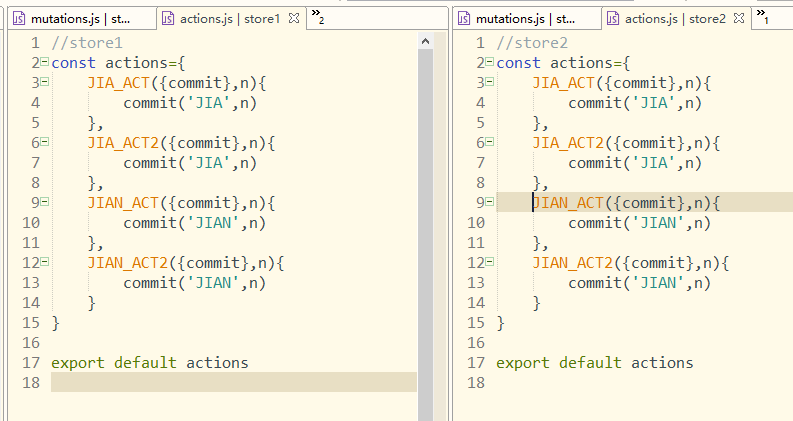
State



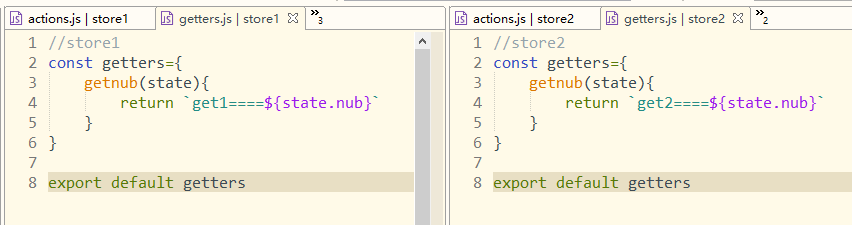
Mutations



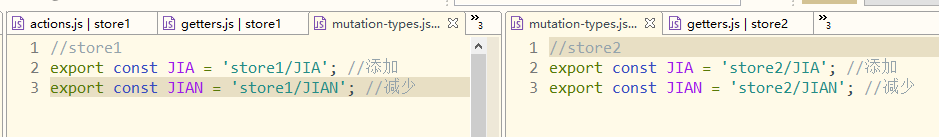
Actions



Getters



Types



## 用法：

### State Getters

#### 方法一：

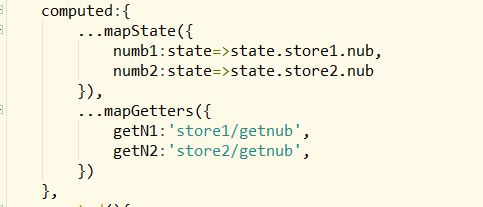
template中直接写入

state：{{$store.state.模块名称.state中的key}}

getters：{{this.$store.getters['模块名称/方法名']}}

#### 方法二：

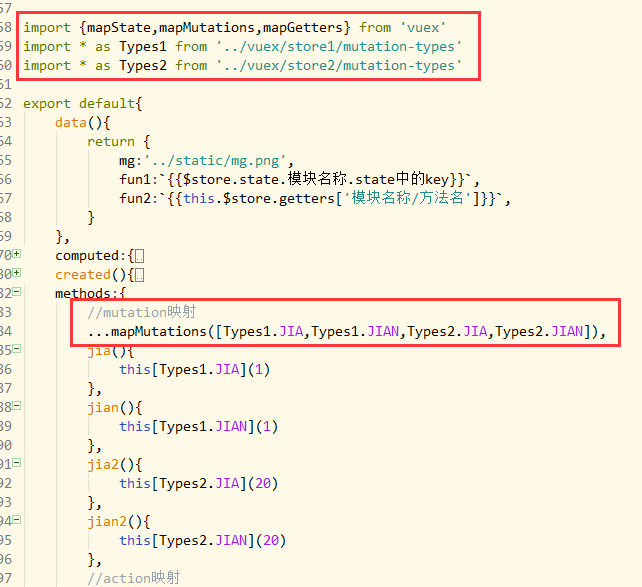
映射到computed



### Mutations

1 引入各模块的types，总mutations

2 映射到method



### Actions

1 引入createNamespacedHelpers

2 引入格模块中的actions

3 映射到method

