原文：<https://www.cnblogs.com/chinabin1993/p/9848720.html>

**先说一下vuex到底是什么？**

vuex 是一个专门为vue.js应用程序开发的状态管理模式。

这个状态我们可以理解为在data中的属性，需要多个组件之间共享的部分。

**vuex中，有默认的五种基本的对象：**

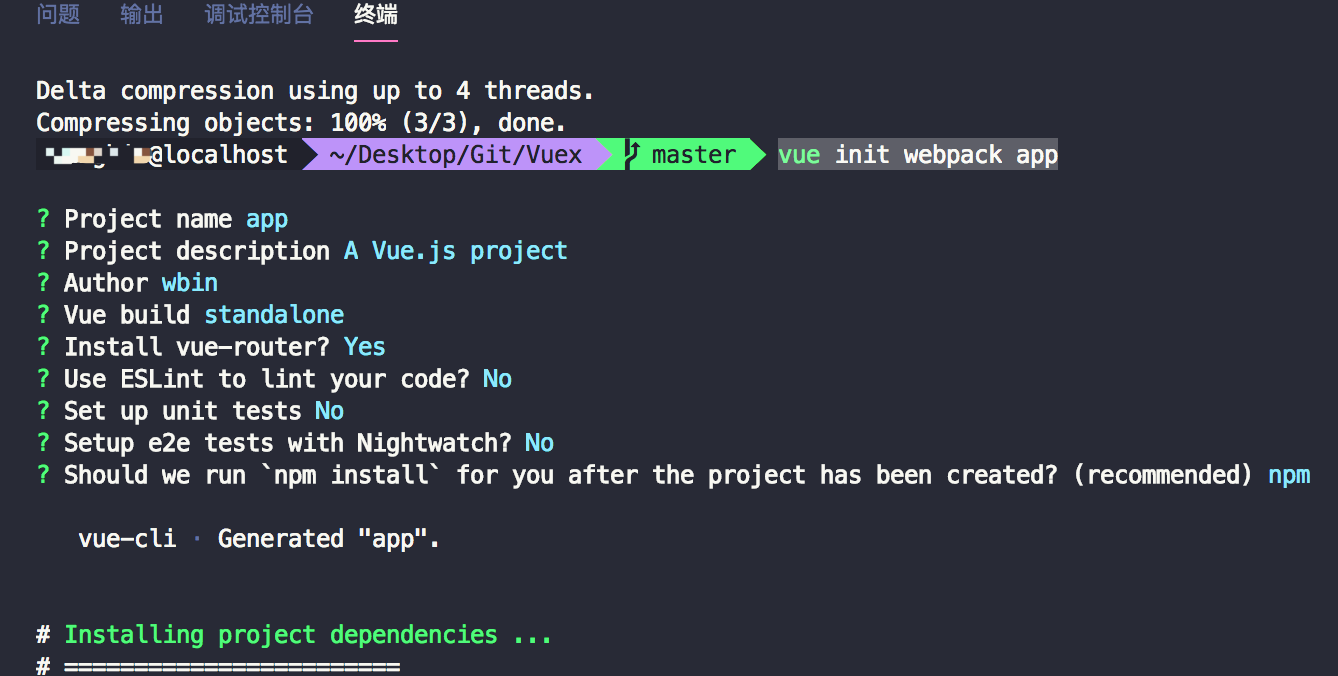
* state：存储状态（变量）
* getters：对数据获取之前的再次编译，可以理解为state的计算属性。我们在组件中使用 $sotre.getters.fun()
* mutations：修改状态，并且是同步的。在组件中使用$store.commit('',params)。这个和我们组件中的自定义事件类似。
* actions：异步操作。在组件中使用是$store.dispath('')
* modules：store的子模块，为了开发大型项目，方便状态管理而使用的。这里我们就不解释了，用起来和上面的一样。

下面我们正式开始，一步步使用vuex

**1、首先创建一个vue-cli项目**

执行下面的命令，创建一个app项目

vue init webpack app



**2、创建完成之后，我们进入文件夹下，并且运行项目**

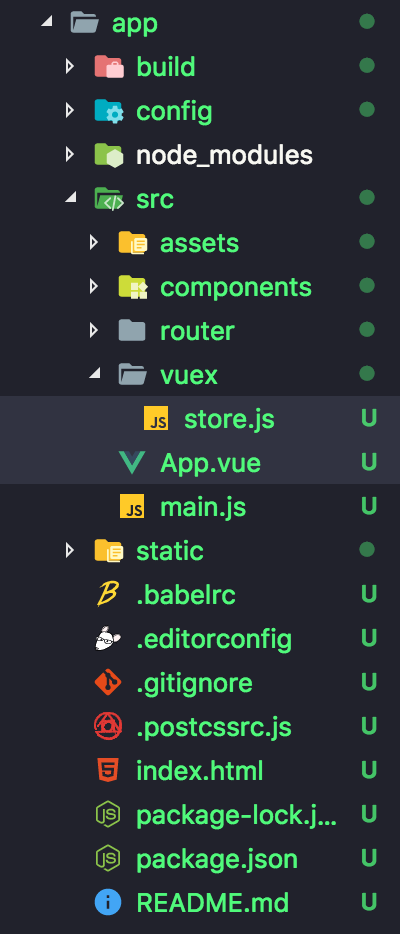
cd app/

npm run dev

接下来我们在src目录下创建一个vuex文件夹

并在vuex文件夹下创建一个store.js文件

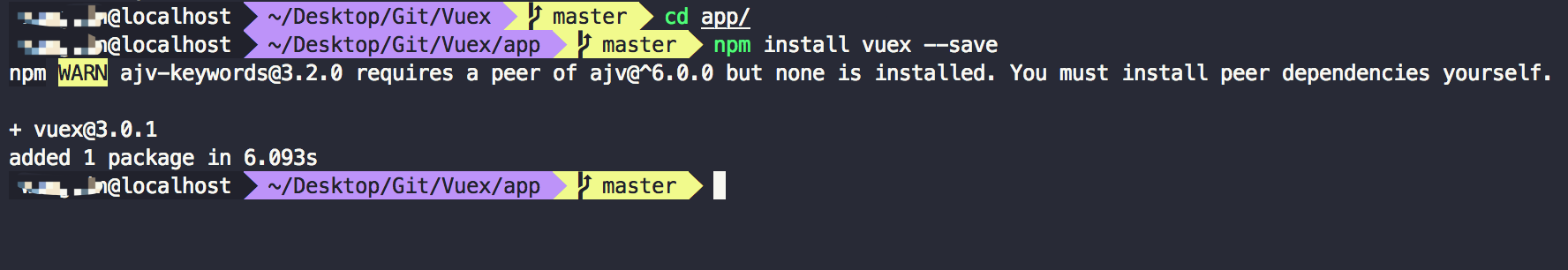
文件夹目录长得是这个样子



**3、目前我们还没有引入vuex，我们需要先下载vuex，并且引入它**

在保证我们处于我们项目下，在命令行输入下面命令，安装vuex

npm install vuex --save



**4、安装成功之后，我们就可以在store.js中尽情玩耍我们的vuex了！**

在store.js文件中，引入vuex并且使用vuex，这里注意我的变量名是大写Vue和Vuex

import Vue from 'vue'

import Vuex from 'vuex'

Vue.use(Vuex)

const state = {

    count: 0

}

export default new Vuex.Store({

    state

})

接下来，在main.js中引入store

import Vue from 'vue'

import App from './App'

import router from './router'

import store from './vuex/store' // 引入store

Vue.config.productionTip = false

/\* eslint-disable no-new \*/

new Vue({

    el: '#app',

    router,

    store,

    components: { App },

    template: '<App/>'

})

然后我我们在任意一个组件中就可以使用我们定义的count属性了。

这里我们在helloWorld.vue中使用一下

<template>

  <div class="hello">

    <h3>{{$store.state.count}}</h3>

  </div>

</template>

现在我们已经使用了vuex中的state，接下来我们如何操作这个值呢？没错！用mutations和actions

我们继续操作store.js文件

我们在sotre.js中定义mutations对象，该对象中有两个方法，mutations里面的参数，第一个默认为state，接下来的为自定义参数。

我们在mutations中定义两个方法，增加和减少，并且设置一个参数n，默认值为0，然后在Vuex.Store中使用它

/\*\*

 \* mutations 里面放置的是我们操作state对象属性的方法

 \*/

const mutations = {

    mutationsAddCount(state, n = 0) {

        return (state.count += n)

    },

    mutationsReduceCount(state, n = 0) {

        return (state.count -= n)

    }

}

export default new Vuex.Store({

    state,

    mutations

})

然后我们在helloWorld.vue中，使用这个方法

还记得我们如何在组件中使用mutations吗？就和自定义事件非常相似

<template>

  <div class="hello">

    <h3>{{$store.state.count}}</h3>

    <div>

      <button @click="handleAddClick(10)">增加</button>

      <button @click="handleReduceClick(10)">减少</button>

    </div>

  </div>

</template>

methods: {

    handleAddClick(n){

        this.$store.commit('mutationsAddCount', n);

    },

    handleReduceClick(n){

        this.$store.commit('mutationsReduceCount', n);

    }

}

接下来就是actions

actions是异步操作

创建actions对象，并且使用

这里我在两个方法中使用了两个不同的参数，一个是context，它是一个和store对象具有相同对象属性的参数。在第二个函数中，我是直接使用了这个对象的commit的方法。

凭大家喜好就行

const actions = {

    actionsAddCount(context, n = 0) {

        console.log(context)

        return context.commit('mutationsAddCount', n)

    },

    actionsReduceCount({ commit }, n = 0) {

        return commit('mutationsReduceCount', n)

    }

}

export default new Vuex.Store({

    state,

    mutations,

    actions

})

在helloWorld.vue中

在methods中，增加两个方法，使用dispath来触发

<template>

  <div class="hello">

    <div>异步操作</div>

    <div>

      <button @click="handleActionsAdd(10)">异步增加</button>

      <button @click="handleActionsReduce(10)">异步减少</button>

    </div>

  </div>

</template>

handleActionsAdd(n){

    this.$store.dispatch('actionsAddCount', n)

},

handleActionsReduce(n){

    this.$store.dispatch('actionsReduceCount', n)

}

最后就是getters

我们一般使用getters来获取我们的state，因为它算是state的一个计算属性

const getters = {

    getterCount(state, n = 0) {

        return (state.count += n)

    }

}

export default new Vuex.Store({

    state,

    mutations,

    actions,

    getters

})

<h4>{{count}}</h4>

const getters = {

    getterCount(state) {

        return (state.count += 10)

    }

}