链滴

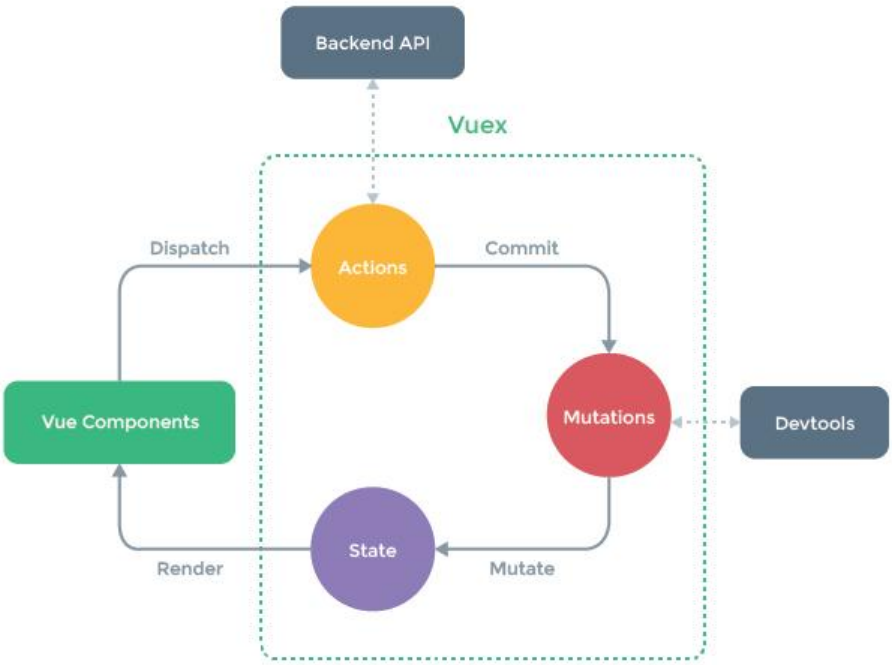
从源码来阅读 Vuex

作者：gmw-zjw
原文链接：https://ld246.com/article/1571387582097
来源网站：链滴

今天来整理下vuex的，由于都是自己理解，哪里出错还希望各位指出！

首先来聊聊 Vuex 与 Redux 的，首先他们都是基于 Flux 的实现，Vuex只是针对 Vue量身开发的，而edux不止单单在React上可使用。

首先对Vuex的API你必须懂，最起码会使用，看下面这张图：



接下来我们翻看看源码：

原文链接：从源码来阅读 Vuex

入口：

```
// mixin.js
export default function (Vue) {

// 省略非关键代码

/*Vuex的init钩子，会存入每一个Vue实例等钩子列表*/
  function vuexInit () {
    // this.$options就是vm实例上的$options
    const options = this.$options
    // store injection
    // 判断vm options上是否有store实例
    if (options.store) {
      // 判断是否是root节点、如果是 那么直接执行 function
      this.$store = typeof options.store === 'function'
        ? options.store()
        : options.store
    } else if (options.parent && options.parent.$store) {
      // 子组件直接从父组件获取store，这就保证了所有组件只有一个store，也就是单一数据流, 保证
态的唯一特性
      this.$store = options.parent.$store
    }
  }
}
```

Store：

来看看 store 源码:

```
export class Store {
   // 构造函数
  constructor (options = {}) {
    // Auto install if it is not done yet and `window` has `Vue`.
    // To allow users to avoid auto-installation in some cases,
    // this code should be placed here. See #731
    /*
      在浏览器环境下，如果插件还未安装（!Vue即判断是否未安装），则它会自动安装。
      它允许用户在某些情况下避免自动安装。
    */
    if (!Vue && typeof window !== 'undefined' && window.Vue) {
      install(window.Vue)
    }

    // 如果我们不挂在vuex这里就会报错
    if (process.env.NODE_ENV !== 'production') {
      assert(Vue, `must call Vue.use(Vuex) before creating a store instance.`)
      assert(typeof Promise !== 'undefined', `vuex requires a Promise polyfill in this browser.`)
      assert(this instanceof Store, `store must be called with the new operator.`)
    }

    const {
      /* 一个数组，包含应用在 store 上的插件方法。这些插件直接接收 store 作为唯一参数，
       * 可以监听 mutation（用于外部地数据持久化、记录或调试）或者提交 mutation
       *（用于内部数据，例如 websocket 或 某些观察者
```

```
     */
    plugins = [],
    // 是否使用严格模式
    /*使 Vuex store 进入严格模式，在严格模式下，任何 mutation 处理函数以外修改 Vuex state
会抛出错误。*/
    strict = false
  } = options

  // store internal state
  /* 用来判断严格模式下是否是用mutation修改state的 */
  this._committing = false
  // 存放action
  this._actions = Object.create(null)
  // 存放 subsribe
  this._actionSubscribers = []
  // 存放 mutation
  this._mutations = Object.create(null)
  this._wrappedGetters = Object.create(null)
  // 根据namespace存放module
  this._modules = new ModuleCollection(options)
  this._modulesNamespaceMap = Object.create(null)
  // 存放订阅者
  this._subscribers = []
  /* 用来实现Watch的Vue实例 */
  this._watcherVM = new Vue()

  // bind commit and dispatch to self
  /*将dispatch与commit调用的this绑定为store对象本身，否则在组件内部this.dispatch时的this
指向组件的vm*/
  const store = this
  const { dispatch, commit } = this
  /* 为dispatch与commit绑定this（Store实例本身） */
  this.dispatch = function boundDispatch (type, payload) {
    return dispatch.call(store, type, payload)
  }
  this.commit = function boundCommit (type, payload, options) {
    return commit.call(store, type, payload, options)
  }

  // strict mode
  /*严格模式(使 Vuex store 进入严格模式，在严格模式下，任何 mutation 处理函数以外修改 Vuex
state 都会抛出错误)*/
  this.strict = strict

  // 根节点的state状态
  const state = this._modules.root.state

  // init root module.
  // this also recursively registers all sub-modules
  // and collects all module getters inside this._wrappedGetters
  /*初始化根module，这也同时递归注册了所有子module，收集所有module的getter到_wrapped
etters中去，this._modules.root代表根module才独有保存的Module对象*/
  // this 当前vuex实例
  installModule(this, state, [], this._modules.root)
```

```
    // initialize the store vm, which is responsible for the reactivity
    // (also registers _wrappedGetters as computed properties)
    /* 通过vm重设store，新建Vue对象使用Vue内部的响应式实现注册state以及computed */
    resetStoreVM(this, state)

    // apply plugins
    // 插件调用
    plugins.forEach(plugin => plugin(this))

    // 插件
    const useDevtools = options.devtools !== undefined ? options.devtools : Vue.config.devtoo
s
    if (useDevtools) {
      devtoolPlugin(this)
    }
  }
}
```

mapState：

mapState 合并多个state，声明在 computed 中

```
// 合并多个state
// 在computed计算属性中，状态的改变会开销很大，所以要声明在 computed 中
export const mapState = normalizeNamespace((namespace, states) => {
  const res = {}
  // 将所有的state遍历，并添加到res中合并返回
  normalizeMap(states).forEach(({ key, val }) => {
    res[key] = function mappedState () {
      // $store 我们在使用时可以通过vm.$store.xxx 获取
      let state = this.$store.state
      let getters = this.$store.getters
      if (namespace) {
        const module = getModuleByNamespace(this.$store, 'mapState', namespace)
        if (!module) {
          return
        }
        state = module.context.state
        getters = module.context.getters
      }
      return typeof val === 'function'
        ? val.call(this, state, getters)
        : state[val]
    }
    // mark vuex getter for devtools
    res[key].vuex = true
  })
  // console.log(res.key)
  return res
})
```

mapGetters：

getters 派生器，从store中派发状态属性：

```
/**
 * Reduce the code which written in Vue.js for getting the getters
 * @param {String} [namespace] - Module's namespace
 * @param {Object|Array} getters
 * @return {Object}
 */
export const mapGetters = normalizeNamespace((namespace, getters) => {
  const res = {}
  normalizeMap(getters).forEach(({ key, val }) => {
    // The namespace has been mutated by normalizeNamespace
    val = namespace + val
    res[key] = function mappedGetter () {
      // 如果命名空间不是当前的 直接返回
      if (namespace && !getModuleByNamespace(this.$store, 'mapGetters', namespace)) {
        return
      }
      if (process.env.NODE_ENV !== 'production' && !(val in this.$store.getters)) {
        console.error(`[vuex] unknown getter: ${val}`)
        return
      }
      // 返回获取store的值
      return this.$store.getters[val]
    }
    // mark vuex getter for devtools
    res[key].vuex = true
  })
  return res
})
```

mapMutations:

```
/**
 * Reduce the code which written in Vue.js for committing the mutation
 * @param {String} [namespace] - Module's namespace
 * @param {Object|Array} mutations # Object's item can be a function which accept `commit`
unction as the first param, it can accept anthor params. You can commit mutation and do any
other things in this function. specially, You need to pass anthor params from the mapped fun
tion.
 * @return {Object}
 */
export const mapMutations = normalizeNamespace((namespace, mutations) => {
  const res = {}
  normalizeMap(mutations).forEach(({ key, val }) => {
    res[key] = function mappedMutation (...args) {
      // Get the commit method from store
      // 使用commit获取store
      let commit = this.$store.commit
      if (namespace) {
        const module = getModuleByNamespace(this.$store, 'mapMutations', namespace)
        if (!module) {
          return
        }
        commit = module.context.commit
      }
```

```
      return typeof val === 'function'
        ? val.apply(this, [commit].concat(args))
        : commit.apply(this.$store, [val].concat(args))
    }
  })
  return res
})
```