Vuex，从入门到入门

世家 · 1 年前

Vuex 是什么？

官方是这么说的：Vuex 是一个专为 Vue.js 应用程序开发的状态管理模式。它采用集中式存储管理应用的所有组件的状态，并以相应的规则保证状态以一种可预测的方式发生变化。

不懂？呵呵，没关系。我是这么认为的：Vuex 就是前端为了方便数据的操作而建立的一个”前端数据库“。且听下文分解。。。

模块间是不共享作用域的，那么B 模块想要拿到 A 模块的数据，我们会怎么做？不要去想什么”状态管理“、”vuex“、”redux“、”函数式编程“什么的。。。

我们会定义一个全局变量，叫
aaa 吧，就是
window.aaa。然后把A 模块要共享的数据作为属性挂到 B 模块上。这样我们在 B 模块中通过 window.aaa 就可以拿到这个数据了。

但是问题来了，B 模块拿到了共享的数据，就叫他
xxx 吧？得了，名字太混乱了，咱先给它们都改下名字。那个全局变量既然是存东西的，就叫
store 吧，共享的数据就叫
state 吧，你叫他
data 也一样，反正我是叫它
state 了。

继续说我们的问题，问题是什么呢？B 模块拿到了 A 模块的数据
state，但是这个数据不是一成不变的呀，A 要操作这个数据的。那么我们是不是要在这个数据——state 改变的时候通知一下 B？那写个自定义事件吧。。。

好吧，你自己已经实现了一个迷你版的
vuex，其实
vuex 就帮你做了这点事儿。我们来看下它里面都有什么玩意儿。

```
export default new Vuex.Store({
  state,
  getters,
  actions,
  mutations,
})
```

这就是一个典型的
使用vuex生成的仓库。不懂？呵呵，没关系，也没打算说它。我们聊点别的。

后端是干什么的？进行数据库操作，处理请求，根据请求分发响应。我们就聊聊数据库的操作，不说 API。首先，你得能取吧？那么得有一套取数据的 API，我们给他们集中起个名字吧？既然是获取，那就叫getter 吧。我们还得存数据呀，是吧。存数据就是对数据库的修改，这些 API，我们也得给它起个名字，就叫
mutation，就这么定了。

OK，vuex 生成的仓库也就这么出来了，所以我说 vuex 就是” 前端的数据库“。State 就是数据库。Mutations 就是我们把数据存入数据库的 API，用来修改
state 的。getters 是我们从数据库里取数据的 API，既然是取，那么你肯定不能把数据库给改了吧？所以
getters 得是一个”纯函数“，就是不会对原数据造成影响的函数，比如
数组的concat（）方法、slice（）方法；与之对应的是数组的
push（）方法、splice（）方法，他们会改变原数组的值。然后我们把这几部分用
store 包一下，” vuex “就捣置出来了，用数据了就从
state 中取，改数据了记得
mutation 到 state 中。

哎，还漏了个
actions 呢。你想呀，后端从前端拿到了数据，总要做个处理吧，处理完了再存到数据库中。其实这就是
action的过程。当然你也可以不做处理，直接丢到数据库，所以vuex也可以在
action 中直接存，就是直接mutation。

现在，我们再来看看 vuex 的数据流。后（qian）端通过
action处理数据，然后通过
mutation 把处理后的数据放入数据库（state）中，谁要用就通过
getter从数据库（state）中取。

你都理解了？好吧，有卵用，还得回归到 API 上。

1、getters

```
// 获取控制变量 ctrl
export function getShowPage (state) {
  return state.ctrl.showPage
}
```

getters 是一个纯函数，接收参数 state，返回你想取的值，都不需要贴数据结构，很清晰吧。

```
// 获取store各项信息
export function getMeta (state) {
  return state.meta
}
```

这张图上，我们返回的是一个被变量控制的值，还可以先对获得的数据进行处理，然后再返回出去。

2、mutations

```
// 公共控制变量 ctrl
```

```
  [SHOW_PAGE] (state) {
    state.ctrl.showPage = true
  },
```

一个最简单的
mutation，就是把
state.ctrl.showData 的值改成 true，好吧，我觉得我是在说废话。Mutation除了接收
state 作为第一个参数外，还可以接收其他的参数，比如：

```
 [NEW_DATA] (state, payload, id){
    const newData = {id, data: payload}
    state.meta = Object.assign({}, {currentData: id})
    state.datas = Object.assign({}, newData)
  },
```

不用说了吧。。。

3、store与state

state就是根据你项目的需求，自己定义一个数据结构。Store中至少要注入两项，state 和
mutation。

```
const state = {
  currentPage: 1,
  user: '0121213',
  change: 0,
  page,
  ctrl,
  meta,
  configs,
  datas
}

export default new Vuex.Store({
  state,
  mutations,
})
```

这是我定义的
state 和 store 的注入。

4、action

先来一个简单点的。

```
export const updateName = function({ dispatch, state }, name) {
  const payload = {name}
  dispatch('UPDATE_PAGE', payload)
}
```

一个叫
updateName 的 action（数据的中间处理），前面花括号是一个参数，state，如果不理解，
看看阮一峰的 ES6入门，对象解构。它接收用户输入的数据
name，然后中间处理的过程就是把
name 包成了对象{name: name}，然后通过
mutation（update_page）
存储。至于 update_page 干了什么，那不是
action（中间处理）的事儿，那是
mutation（存储）的逻辑。

```
export const updateData = function({ dispatch, state }, data) {
  const payload = data
  const id = state.meta.currentData
  if (id === 'initial') {
    const id = createDataId()
    dispatch('NEW_DATA', payload, id)
  } else {
    dispatch('UPDATE_DATA', payload)
  }
}
```

这个 action 叫 updateData,里面先把 data 改了个名字，然后加了判断，如果 id
是 ‘initial’
就怎么存。。。否则怎么存。。。

再来一个：

```
export const update = function({ dispatch, state }, payload) {
  const id = state.meta.current
  if (id === 'initial') {
    const id = createChartId()
    dispatch('NEW', payload, id)
  } else {
    dispatch('UPDATE', payload)
  }

  var privateConfig = {}, initialData = []
  const cfgId = state.meta.currentConfig
  const dtId = state.meta.currentConfig

  switch (payload.format) {∞


  if (cfgId === 'initial') {
    const cfgId = createConfigId()
    dispatch('NEW_CONFIG', privateConfig, cfgId)
  } else {
    dispatch('UPDATE_CONFIG', privateConfig)
  }

  if (dtId === 'initial') {
    const dtId = createDataId()
    dispatch('NEW_DATA', initialData, dtId)
  }
}
```

这个 action 很长，被我收起来了，我们看看它都做了什么。接收一个参数 payload，判断如

果 id 是 initial，就生成
id，然后怎么存。。。如果
id 不是 initial，就怎么存。。。好像都一个套路。接着看，定义空的”
privateConfig “、” initialData “，switch 块里面就是生成 privateConfig、initialData的地方。然后判断如果
cfid 是 initial，就生成
id，然后怎么存。。。如果
cfid 不是 initial，就怎么存。。。好吧，全是套路。

通过这几个小例子，我希望大家可以看到的是，各个类型的 API各司其职，mutation 只管存，你给我（dispatch）我就存；action只管中间处理，处理完我就给你，你怎么存我不管；Getter 我只管取，我不改的。

咱们再来看一个听着玄乎其神的异步操作的 action。

```
export const save = function({ dispatch, state }) {
  const params = state
  api.save(params).then(function(res) {
    console.log(res)
    dispatch('SAVE_SUCCESS', res)
  }).catch((err) => {
    console.log(err)
    dispatch('SAVE_FAIL', err)
  })
}
```

一个叫 save 的 action。首先在里面发起了一个请求，这个请求是经过包装的（改了名字叫 api.save），咱不管它改名字这茬儿。第一个
then 就是成功的回调，通过
res 拿到数据，拿到数据了就怎么存。。。第二个
catch 就是失败的回调，通过
err 拿到错误信息，拿到错误信息了怎么存。。。呵呵，跟if判断差不多嘛。。。其实异步的 action 还可以发送第三个
dispatch 的，在发起请求前先保存下原始数据，有时候有需求会用到的，比如官方 DEMO 的购物车。

看来形式怎么变，action 里面的逻辑不会变，它做的事情不会变。

5、它们是什么？

至于 mutation_type，mudules。。。这些都不是重点，可用可不用。Vuex 最强大的地方还在于它的伸缩性。你项目大了，需要分
mudules，项目小了一样可以用呀，甚至只需要建立一个
store.js，你有什么理由不用它？

理解了 getter，mutation，action了，但是也许你还不知道该怎么去用，因为你还不知道他们是什么东西。

```
import { mapGetters } from 'vuex'

export default {
  // ...
  computed: {
  // 使用对象展开运算符将 getters 混入 computed 对象中
    ...mapGetters([
      'doneTodosCount',
      'anotherGetter',
      // ...
    ])
  }
}
```

截的官网的图，重点不是 mapGetters，重点是mapGetters写在了哪儿。写在了 computed 里面，这说明虽然
getter我们写的是函数，但是我们应该把它当成计算属性来用。再来看
action。

```
import { mapActions } from 'vuex'

export default {
  // ...
  methods: {
    ...mapActions([
      'increment' // 映射 this.increment() 为 this.$store.dispatch('increment')
    ]),
    ...mapActions({
      add: 'increment' // 映射 this.add() 为 this.$store.dispatch('increment')
    })
  }
}
```

action放在了 methods 里面，说明我们应该把它当成函数来用。你以前写的 onclick = ‘函数’，现在就可以
onclick = ‘action’。我们在 vue 里面是这样写的，@click
= “action()”，加不加括号看心情。

最后来看看 mutation。

```
const store = new Vuex.Store({
  state: {
    count: 1
  },
  mutations: {
    increment (state) {
      // 变更状态
      state.count++
    }
  }
})
```

它是在 store 里面写的，这说明，它就是个半成品，中间量，我们不应该在外面去操作它。

Vuex 什么时候用？

如果你问了这个问题，说明你不该用。但这个问题可以很容易并且很正确的回答。

等你痛了的时候，你就该用了。如果你连自己的代码都看不懂的时候，如果你自己都搞不清楚值在组件中是怎么传递的时候，如果你自己代码写了一半，恶心的想要撂挑子不干的时候，赶紧的，Vue.use（Vuex）！

看完文章,多翻翻评论,里面都是小朋友们在开发中遇到的问题,比文章更有价值...

Vue.js Vuex

58 条评论

写下你的评论...

小萌
终于找到一篇通俗易懂的vuex了，感谢作者。
1 年前 4 赞

世家（作者）
哈哈，只是在使用过程中感觉应该是这样的。。。估计被尤大看到了要打我的
1 年前

小萌
最后那个问题不是到底什么时候使用vuex管理状态合适，管理全局状态？这个状态可以在锁头组件中长驱直入的，就不要在组件内部对这些状态进行管理吗？
1 年前

世家（作者）回复 小萌 查看对话
我是觉得，只要你理解了vuex，那么不管大小的项目都应该去使用它，因为项目中必然会有需要持久化的数据。。。项目小了，建一个store.js的文件就成了，不麻烦。
1 年前

Vancover
很详细！！
1 年前

高明
这篇是讲vuex得最好的文章了。抛弃所有没用的概念，用数据库这种类比，一下让人明白vuex是什么，能用来干什么，什么时候用。感谢！
1 年前 2 赞

世家（作者）回复 高明 查看对话
嘿嘿，谢谢
1 年前

辰土
请教一个困惑，实际用的时候，前端要使用state，但state里的数据只是临时的。当state变化时，有些信息比如草稿，要存储在本地；比如新增一个数据，要更新到服务器。这个时候是应该用action在state变更前就异步推送到服务，然后再dispatch吗？ 就是说 与真正的数据库的交互应用用action且在写入state之前操作？而不是 “监控state变化，再处理”这种逻辑？
1 年前 2 赞

世家（作者）回复 辰土 查看对话
额，state有两个，一个是组件内部state，它负责组件私有状态。还有一个state是vuex维护的state。这是state是所有组件所共用的。这两个state的变化，vue都能监视到，从而更新视图。所以组件私有的state应该写到state上；而公共state，需要持久化的state需要挂到state上!哈哈，等你看懂这句话的时候，我觉得你的疑惑就解决了。
1 年前

和安喜
感谢作者，通俗易懂，很容易明白。谢谢。高手就是能把复杂的东西简洁明了的表述出来。非常感谢
1 年前