Vuex面试整理

基本概念

state

> 单一状态树。

getters

> Vuex 允许我们在 store 中定义“getter”（可以认为是 store 的计算属性）。就像计算属性一样，getter 的返回值会根据它的依赖被缓存起来，且只有当它的依赖值发生了改变才会被重新计算。
>
> Getter 接受 state 作为其第一个参数：

```
const store = new Vuex.Store({
state: {
 todos: [
   { id: 1, text: '...', done: true },
   { id: 2, text: '...', done: false }
 ]
},
getters: {
 doneTodos: state => {
   return state.todos.filter(todo => todo.done)
 }
}
})
```

action

> Action 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。

mutation

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

用于Vue

dispatch

> Action 通过 `store.dispatch` 方法触发：

```
store.dispatch('increment')
```

commit

> 以相应的 type 调用 **store.commit** 方法：

```
store.commit('increment')
```

你可以向 `store.commit` 传入额外的参数，即 mutation 的 **载荷（payload）**：

```
// ...
mutations: {
increment (state, n) {
 state.count += n
}
}
store.commit('increment', 10)
```

在大多数情况下，载荷应该是一个对象，这样可以包含多个字段并且记录的 mutation 会更易读：

```
// ...
mutations: {
increment (state, payload) {
 state.count += payload.amount
}
}
store.commit('increment', {
amount: 10
})
```

提交 mutation 的另一种方式是直接使用包含 `type` 属性的对象：

```
store.commit({
type: 'increment',
amount: 10
})
```

当使用对象风格的提交方式，整个对象都作为载荷传给 mutation 函数，因此 handler 保持不变：

```
mutations: {
increment (state, payload) {
 state.count += payload.amount
}
}
```

mapState

当一个组件需要获取多个状态时候，将这些状态都声明为计算属性会有些重复和冗余。为了解决这个问题，我们可以使用 `mapState` 辅助函数帮助我们生成计算属性让你少按几次键：

```
// 在单独构建的版本中辅助函数为 Vuex.mapState
import { mapState } from 'vuex'

export default {
// ...
computed: mapState({
 // 箭头函数可使代码更简练
```

```
  count: state => state.count,

  // 传字符串参数 'count' 等同于 `state => state.count`
  countAlias: 'count',

  // 为了能够使用 `this` 获取局部状态，必须使用常规函数
  countPlusLocalState (state) {
    return state.count + this.localCount
  }
})
}
```

当映射的计算属性的名称与 state 的子节点名称相同时，我们也可以给 `mapState` 传一个字符串数组。

```
computed: mapState([
// 映射 this.count 为 store.state.count
'count'
])
```

mapGetters

`mapGetters` 辅助函数仅仅是将 store 中的 getter 映射到局部计算属性：

```
import { mapGetters } from 'vuex'

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

如果你想将一个 getter 属性另取一个名字，使用对象形式：

```
mapGetters({
// 把 `this.doneCount` 映射为 `this.$store.getters.doneTodosCount`
doneCount: 'doneTodosCount'
})
```

mapActions

在组件中使用 `this.$store.dispatch('xxx')` 分发 action，或者使用 `mapActions` 辅助函数将组件的 methods 映射为 `store.dispatch` 调用（需要先在根节点注入 `store`）：

```
import { mapActions } from 'vuex'

export default {
// ...
```

```
methods: {
 ...mapActions([
   'increment', // 将 `this.increment()` 映射为
`this.$store.dispatch('increment')`

   // `mapActions` 也支持载荷：
   'incrementBy' // 将 `this.incrementBy(amount)` 映射为
`this.$store.dispatch('incrementBy', amount)`
 ]),
 ...mapActions({
   add: 'increment' // 将 `this.add()` 映射为
`this.$store.dispatch('increment')`
 })
}
}
```

mapMutations

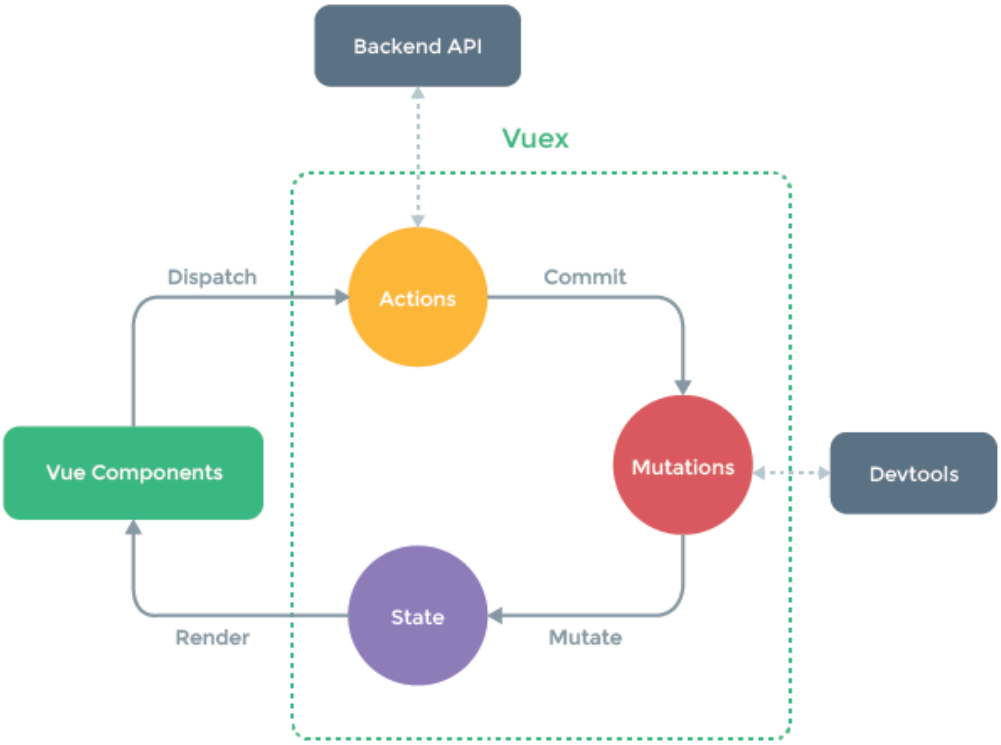
> 可以在组件中使用 `this.$store.commit('xxx')` 提交 mutation，或者使用 `mapMutations` 辅助函数将组件中的 methods 映射为 `store.commit` 调用（需要在根节点注入 `store`）。

```
import { mapMutations } from 'vuex'

export default {
// ...
methods: {
 ...mapMutations([
   'increment', // 将 `this.increment()` 映射为
`this.$store.commit('increment')`

   // `mapMutations` 也支持载荷：
   'incrementBy' // 将 `this.incrementBy(amount)` 映射为
`this.$store.commit('incrementBy', amount)`
 ]),
 ...mapMutations({
   add: 'increment' // 将 `this.add()` 映射为
`this.$store.commit('increment')`
 })
}
}
```

原理图



注意：异步操作只能在actions中操作