ql_vuex

首先利用脚手架创建项目

```
1. vue create ql_vuex
2. Manually select features
3. 选择Babel Vuex
4. 2.x
5. In dedicated config files
6. N
```

实现在Store中定义的数据显示在页面上的两种方式

store/index.js

```
import Vue from 'vue'
import Vuex from 'vuex'

Vue.use(Vuex)

//2 每一个vue实例中都有一个属性$store
export default new Vuex.Store({
  state: {
    //1. 存储数据，怎么将num展示在页面上呢？
    num: 1
  },
  //2.2 方式二：添加一个getter，在这里面可以添加一个方法
  getters: {
    // 只读，并且可以对数据进行处理
    getNum (state) {
      return state.num * 2
    }
  },
  mutations: {
  },
  actions: {
  },
  modules: {
  }
})
```

App.vue

```
<template>
  <div class="app">
    // 2. 怎么显示store state里面的num数据呢?
    // 2.1 第一种方式this.$store.state.num
    <div>第一种方式Num:{{this.$store.state.num}}</div>
    // 2.2第二种方式可以在store中设置getters，这种方式是只读的，并且可以对数据进行处理，这里运用的时候不用写()，写()意味着方法需要传参
    <div>第二种方式Num:{{this.$store.getters.getNum}}</div>
  </div>
</template>
```

```
<script>
export default {

}
</script>

<style scoped>
.app {
  font-size: larger;
  font-weight: bold;
}
</style>
```

按钮实现同步增加减少、mutation里面是同步的代码

store/index.js

```
import Vue from 'vue'
import Vuex from 'vuex'

Vue.use(Vuex)

// 每一个vue实例中都有一个属性$store
export default new Vuex.Store({
  state: {
    // 存储数据，怎么将num展示在页面上呢？
    num: 1
  },
  //2.2 方式二：添加一个getter，在这里面可以添加一个方法
  getters: {
    // 只读，并且可以对数据进行处理 参数是固定的
    getNum (state) {
      return state.num * 2
    }
  },
  mutations: {
    //3. mutations 相当于是同步的方式 参数是固定的
    // payload 是传入的参数
    //3.1 同步增加
    syncIncrement (state, payload) {
      state.num += payload
    },
    //3.1 同步减少
    syncDecrement (state, payload) {
      state.num -= payload
    }
  },
  actions: {
    // actions 相当于是异步的方式
  },
  modules: {
  }
})
```

App.vue

```
<template>
  <div class="app">
    // 2. 怎么显示store state里面的num数据呢?
    // 2.1 第一种方式this.$store.state.num
    <div>第一种方式Num:{{this.$store.state.num}}</div>
    // 2.2第二种方式可以在store中设置getters，这种方式是只读的，并且可以对数据进行处理，这里用的时候不用写()，写()意味着方法需要传参
    <div>第二种方式Num:{{this.$store.getters.getNum}}</div>
    <button @click="increment">同步增加Num</button>
    <button @click="decrement">同步减少Num</button>
  </div>
</template>

<script>
export default {
  name: "app",
  methods: {
    increment () {
      this.$store.commit("syncIncrement", 10)
    },
    decrement () {
      this.$store.commit("syncDecrement", 5)
    }
  }
}
</script>

<style scoped>
.app {
  font-size: larger;
  font-weight: bold;
}
</style>
```

在actions里面异步实现增加操作

store/index.js

```
import Vue from 'vue'
import Vuex from 'vuex'  // 引用第三方库


Vue.use(Vuex)  // 使用插件，插件插入vue里面才能使用

// 每一个vue实例中都有一个属性$store
export default new Vuex.Store({
  state: {
    // 存储数据，怎么将num展示在页面上呢?
    num: 1
  },
  //2.2 方式二：添加一个getter，在这里面可以添加一个方法
  getters: {
    // 只读，并且可以对数据进行处理 参数是固定的
    getNum (state) {
      return state.num * 2
    }
  },
```

```
  mutations: {
    //3. mutations 相当于是同步的方式 参数是固定的
    // payload 是传入的参数
    //3.1 同步增加
    syncIncrement (state, payload) {
      state.num += payload
    },
    //3.1 同步减少
    syncDecrement (state, payload) {
      state.num -= payload
    }
  },
  actions: {
    //4. actions 相当于是异步的方式
    //4.1 异步增加
    asyncIncrement ({ commit, dispatch }, payload) {
      // 模拟ajax
      setTimeout(() => {
        commit("syncIncrement", payload)
      }, 1000)
    }
  },
  modules: {
  }
})
```

App.vue

```
<template>
  <div class="app">
    // 2. 怎么显示store state里面的num数据呢?
    // 2.1 第一种方式this.$store.state.num
    <div>第一种方式Num:{{this.$store.state.num}}</div>
    // 2.2第二种方式可以在store中设置getters，这种方式是只读的，并且可以对数据进行处理，这里运用的时候不用写()，写()意味着方法需要传参
    <div>第二种方式Num:{{this.$store.getters.getNum}}</div>
    <button @click="increment">同步增加Num</button><br>
    <button @click="decrement">同步减少Num</button><br>
    <button @click="asyncIncrement">异步增加Num</button>
  </div>
</template>

<script>
export default {
  name: "app",
  methods: {
    increment () {
      this.$store.commit("syncIncrement", 10)
    },
    decrement () {
      this.$store.commit("syncDecrement", 5)
    },
    //4.2 异步增加
    asyncIncrement () {
      this.$store.dispatch("asyncIncrement", 10)
    }
  }
```

```
}
</script>

<style scoped>
.app {
  font-size: larger;
  font-weight: bold;
}
</style>
```

实现qlvuex

使用方式不变，引入qlvuex

store/index.js

```
import Vue from 'vue'
// import Vuex from 'vuex'  // 引用第三方库
import Vuex from './qlvuex' //5. 使用自己实现的qlvuex
//5.1 注释之后发现页面报错没有找到$store，那么第一步给每个组件都添加一个$store
// 组件中Store添加顺序：先给根对象赋$store然后给子赋值的时候，判断父有没有，如果有就拿到子中
见ql_vuex.drawio

Vue.use(Vuex)  // 使用插件，插件插入vue里面才能使用

// 每一个vue实例中都有一个属性$store
export default new Vuex.Store({
  state: {
    // 存储数据，怎么将num展示在页面上呢？
    num: 1
  },
  //2.2 方式二：添加一个getter，在这里面可以添加一个方法
  getters: {
    // 只读，并且可以对数据进行处理 参数是固定的
    getNum (state) {
      return state.num * 2
    }
  },
  mutations: {
    //3. mutations 相当于是同步的方式 参数是固定的
    // payload 是传入的参数
    //3.1 同步增加
    syncIncrement (state, payload) {
      state.num += payload
    },
    //3.1 同步减少
    syncDecrement (state, payload) {
      state.num -= payload
    }
  },
  actions: {
    //4. actions 相当于是异步的方式
    //4.1 异步增加
    asyncIncrement ({ commit, dispatch }, payload) {
      // 模拟ajax
      setTimeout(() => {
```

```
        commit("syncIncrement", payload)
      }, 1000)
    }
  },
  modules: {
  }
})
```

store/qlvuex.js

```
// 实现qlvuex

//2. 这里Store是用于做数据存储的
class Store {
    constructor() {

    }
}

//1. 在vue里面使用插件，一定要有这样一个方法，用于安装插件
// 安装插件目的是让每个组件都有一个 $store
const install = () => {

}

//3. 导出
export default {
    Store,
    install
}
```

App.vue

```
<template>
  <div class="app">
    // 2. 怎么显示store state里面的num数据呢?
    // 2.1 第一种方式this.$store.state.num
    <div>第一种方式Num:{{this.$store.state.num}}</div>
    // 2.2第二种方式可以在store中设置getters，这种方式是只读的，并且可以对数据进行处理，这里运用的时候不用写()，写()意味着方法需要传参
    <div>第二种方式Num:{{this.$store.getters.getNum}}</div>
    <button @click="increment">同步增加Num</button><br>
    <button @click="decrement">同步减少Num</button><br>
    <button @click="asyncIncrement">异步增加Num</button>
  </div>
</template>

<script>
export default {
  name: "app",
  methods: {
    increment () {
      this.$store.commit("syncIncrement", 10)
    },
    decrement () {
      this.$store.commit("syncDecrement", 5)
```

```
    },
    //4.2 异步增加
    asyncIncrement () {
      this.$store.dispatch("asyncIncrement", 10)
    }
  }
}
</script>

<style scoped>
.app {
  font-size: larger;
  font-weight: bold;
}
</style>
```

如果我们将原来的vuex变成上面的qlvuex，那么就会报错

解决第一个报错Store错误

```
TypeError: _qlvuex__WEBPACK_IMPORTED_MODULE_0___default(...).Store is not a
constructor
```

store/qlvuex.js 1~12

```
// 实现qlvuex

let Vue

//2. 这里Store是用于做数据存储的
class Store {
    constructor() {

    }
}

//1. 在vue里面使用插件，一定要有这样一个方法，用于安装插件
// 安装插件目的是让每个组件都有一个 $store 1~12
// 3. 这里默认有一个参数 安装插件的时候会把Vue实例传过来
const install = (_Vue) => {
    Vue = _Vue;
    //4.这里有个mixin方法 给每一个组件都注册一个方法，beforeCreate
    Vue.mixin({
        //5. 这里写了之后意味着每个组件都有beforeCreate这样生命周期的一个方法
        beforeCreate () {
            //6. 每个组件注册的时候都会有个日志
            // console.log(this)
            //7. 如果不容易找到那么就在main.js里面加个名字 root 方便判断是那个节点先出来
            // console.log(this.$options)
            //8. 先出来根节点root 后app
            console.log(this.$options.name)
            //9. 如果说这个组件存在，并且上面有store，在main.js里面传入了一个store，
store的入口是在根组件上
            if (this.$options && this.$options.store) {
                //10. 那么就可以说明这个一定是根节点
                this.$store = this.$options.store
            } else {
```

```
                //11. 子组件  如果父组件存在，并且父组件中也存在 store
                this.$store = this.$parent && this.$parent.$store
                //12. 到这一步 store的报错就解决了，那么还剩下一个num的错误
                //Error in render: "TypeError: Cannot read properties of
undefined (reading 'num')"
            }
        }
    })
}

//3. 导出
export default {
    Store,
    install
}
```

解决报错num错误

```
Error in render: "TypeError: Cannot read properties of undefined (reading
'num')"
```

store/qlvuex.js 13-17

```
// 实现qlvuex

let Vue

//2. 这里Store是用于做数据存储的
class Store {
    //13. 这里需要看Store是在哪里new的，是在index.js里面new的时候传入了一个option配置项过
来
    // 这里接收所有的配置项
    constructor(options) {
        //14. 如何接收，这里可以定义一个属性，可以拿到options里面的值，但是最好不要这样写
        // this._s = options.state;
        //15. 因为这个数据需要是响应式的，发生改变之后需要通知页面更新，那么这个数据必须是响应
式的
        //16. 如何变成响应式的数据呢，只需要添加进new Vue中即可
        this.vm = new Vue({
            data: {
                state: options.state
            }
        })
    }
    //17. 这个数据如何访问呢？添加一个get方法，那么页面上第一种方式获取数据就可以显示了
    get state () {
        // 这里返回state对象，那么就可以使用this.$store.state.num了
        return this.vm.state
    }
}

//1. 在vue里面使用插件，一定要有这样一个方法，用于安装插件
// 安装插件目的是让每个组件都有一个 $store 1~12
// 3. 这里默认有一个参数 安装插件的时候会把Vue实例传过来
const install = (_Vue) => {
    Vue = _Vue;
```

```
    //4.这里有个mixin方法 给每一个组件都注册一个方法，beforeCreate
    Vue.mixin({
        //5. 这里写了之后意味着每个组件都有beforeCreate这样生命周期的一个方法
        beforeCreate () {
            //6. 每个组件注册的时候都会有个日志
            // console.log(this)
            //7. 如果不容易找到那么就在main.js里面加个名字 root 方便判断是那个节点先出来
            // console.log(this.$options)
            //8. 先出来根节点root 后app
            console.log(this.$options.name)
            //9. 如果说这个组件存在，并且上面有store，在main.js里面传入了一个store，
store的入口是在根组件上
            if (this.$options && this.$options.store) {
                //10. 那么就可以说明这个一定是根节点
                this.$store = this.$options.store
            } else {
                //11. 子组件  如果父组件存在，并且父组件中也存在 store
                this.$store = this.$parent && this.$parent.$store
                //12. 到这一步 store的报错就解决了，那么下一个处理num的错误
                //Error in render: "TypeError: Cannot read properties of
undefined (reading 'num')"
            }
        }
    })
}

//3. 导出
export default {
    Store,
    install
}
```

```
// 上面13~17步骤的代码写完之后，就可以使用第一种方式获取数据了
<div>第一种方式Num:{{this.$store.state.num}}</div>
```

解决报错getNum方法错误

```
// 但是没有办法用第二种方式获取数据，不能使用getNum方式获取数据
[Vue warn]: Error in render: "TypeError: Cannot read properties of undefined
(reading 'getNum')"
```

store/qlvuex.js 18~24

```
// 实现qlvuex

let Vue

//2. 这里Store是用于做数据存储的
class Store {
    //13. 这里需要看Store是在哪里new的，是在index.js里面new的时候传入了一个option配置项过
来
    // 这里接收所有的配置项
    constructor(options) {
        //14. 如何接收，这里可以定义一个属性，可以拿到options里面的值，但是最好不要这样写
        // this._s = options.state;
```

```
        //15. 因为这个数据需要是响应式的，发生改变之后需要通知页面更新，那么这个数据必须是响应式的
        //16. 如何变成响应式的数据呢，只需要添加进new Vue中即可
        this.vm = new Vue({
            data: {
                state: options.state
            }
        })
        //18.如何解决getNum报错呢？ for getters
        // 在index.js中如果有 getters那么直接拿过来用，如果别人没有写那么设置默认值
        //19. 上面这个是取得是配置项中的getters
        let getters = options.getters || {}
        //20.下面是取的是上面store里面的一个空的属性，因为我们到时候访问是通过
        // this.$store.getters.getNum 访问的，那么肯定就是访问下面的getters
        this.getters = {}
        //21. 把getters中属性定义到this.getters，因为getters里面可能有多个方法，不只getNum一个
        // 那么就需要找到每一个key  getterName是每一个方法的名字
        Object.keys(getters).forEach(getterName => {
            //22. 定义一个属性，把每一个getter(比如getNum)，定义到this.getters ={} 对象中来
            Object.defineProperty(this.getters, getterName, {
                //23. 只有get方法，访问getterName属性就会触发get方法
                get: () => {
                    //24. getters 是对象，getName是函数名，比如这里就是getters[getNum]()取到getNum函数
                    // 并且传入参数 this.state
                    return getters[getterName](this.state)
                }
            })
        })
    }
    //17. 这个数据如何访问呢？添加一个get方法，那么页面上第一种方式获取数据就可以显示了
    get state () {
        // 这里返回state对象，那么就可以使用this.$store.state.num了
        return this.vm.state
    }
}

//1. 在vue里面使用插件，一定要有这样一个方法，用于安装插件
// 安装插件目的是让每个组件都有一个 $store 1~12
// 3. 这里默认有一个参数 安装插件的时候会把Vue实例传过来
const install = (_Vue) => {
    Vue = _Vue;
    //4.这里有个mixin方法 给每一个组件都注册一个方法，beforeCreate
    Vue.mixin({
        //5. 这里写了之后意味着每个组件都有beforeCreate这样生命周期的一个方法
        beforeCreate () {
            //6. 每个组件注册的时候都会有个日志
            // console.log(this)
            //7. 如果不容易找到那么就在main.js里面加个名字 root 方便判断是那个节点先出来
            // console.log(this.$options)
            //8. 先出来根节点root 后app
            console.log(this.$options.name)
            //9. 如果说这个组件存在，并且上面有store，在main.js里面传入了一个store，store的入口是在根组件上
            if (this.$options && this.$options.store) {
                //10. 那么就可以说明这个一定是根节点
```

```
                this.$store = this.$options.store
            } else {
                //11. 子组件  如果父组件存在，并且父组件中也存在 store
                this.$store = this.$parent && this.$parent.$store
                //12. 到这一步 store的报错就解决了，那么还剩下一个num的错误
                //Error in render: "TypeError: Cannot read properties of
undefined (reading 'num')"
            }
        }
    })
}

//3. 导出
export default {
    Store,
    install
}
```

```
// 上面的18~24步骤的代码写完就可以使用第二种方式获取数据
<div>第二种方式Num:{{this.$store.getters.getNum}}</div>
```

解决报错增加操作方法

```
[Vue warn]: Error in v-on handler: "TypeError: this.$store.commit is not a
function"
```

store/qlvuex.js 25

```
// 实现qlvuex

let Vue

//2. 这里Store是用于做数据存储的
class Store {
    //13. 这里需要看Store是在哪里new的，是在index.js里面new的时候传入了一个option配置项过
来
    // 这里接收所有的配置项
    constructor(options) {
        //14. 如何接收，这里可以定义一个属性，可以拿到options里面的值，但是最好不要这样写
        // this._s = options.state;
        //15. 因为这个数据需要是响应式的，发生改变之后需要通知页面更新，那么这个数据必须是响应
式的
        //16. 如何变成响应式的数据呢，只需要添加进new Vue中即可
        this.vm = new Vue({
            data: {
                state: options.state
            }
        })
        //18.如何解决getNum报错呢？ for getters
        // 在index.js中如果有 getters那么直接拿过来用，如果别人没有写那么设置默认值
        //19. 上面这个是取得是配置项中的getters
        let getters = options.getters || {}
        //20.下面是取的是上面store里面的一个空的属性，因为我们到时候访问是通过
        // this.$store.getters.getNum 访问的，那么肯定就是访问下面的getters
        this.getters = {}
```

```
        //21. 把getters中属性定义到this.getters，因为getters里面可能有多个方法，不只
getNum一个
        // 那么就需要找到每一个key  getterName是每一个方法的名字
        Object.keys(getters).forEach(getterName => {
            //22. 定义一个属性，把每一个getter(比如getNum)，定义到this.getters ={} 对
象中来
            Object.defineProperty(this.getters, getterName, {
                //23. 只有get方法，访问getterName属性就会触发get方法
                get: () => {
                    //24. getters 是对象，getName是函数名，比如这里就是
getters[getNum]()取到getNum函数
                    // 并且传入参数 this.state
                    return getters[getterName](this.state)
                }
            })
        })
        // 24. 解决点击事件函数错误 for mutations
        let mutations = options.mutations || {}
        this.mutations = {}
        Object.keys(mutations).forEach(mutationName => {
            this.mutations[mutationName] = payload => {
                // 根据这个mutations[mutationName] 可以找到具体的函数
                mutations[mutationName](this.state, payload);
            }
        })
    }
    //25. 上面写完点击会报错 vue.runtime.esm.js?c320:4603 [Vue warn]: Error in v-on
handler: "TypeError: this.$store.commit is not a function"
    // 没有commit方法，commit传入了 参数的名字和payload
    commit (type, payload) {
        // 这里面已经有方法了
        this.mutations[type](payload)
    }
    //17. 这个数据如何访问呢？添加一个get方法，那么页面上第一种方式获取数据就可以显示了
    get state () {
        // 这里返回state对象，那么就可以使用this.$store.state.num了
        return this.vm.state
    }
}

//1. 在vue里面使用插件，一定要有这样一个方法，用于安装插件
// 安装插件目的是让每个组件都有一个 $store 1~12
// 3. 这里默认有一个参数 安装插件的时候会把Vue实例传过来
const install = (_Vue) => {
    Vue = _Vue;
    //4.这里有个mixin方法 给每一个组件都注册一个方法，beforeCreate
    Vue.mixin({
        //5. 这里写了之后意味着每个组件都有beforeCreate这样生命周期的一个方法
        beforeCreate () {
            //6. 每个组件注册的时候都会有个日志
            // console.log(this)
            //7. 如果不容易找到那么就在main.js里面加个名字 root 方便判断是那个节点先出来
            // console.log(this.$options)
            //8. 先出来根节点root 后app
            console.log(this.$options.name)
            //9. 如果说这个组件存在，并且上面有store，在main.js里面传入了一个store，
store的入口是在根组件上
            if (this.$options && this.$options.store) {
```

```
                //10. 那么就可以说明这个一定是根节点
                this.$store = this.$options.store
            } else {
                //11. 子组件  如果父组件存在，并且父组件中也存在 store
                this.$store = this.$parent && this.$parent.$store
                //12. 到这一步 store的报错就解决了，那么还剩下一个num的错误
                //Error in render: "TypeError: Cannot read properties of
undefined (reading 'num')"
            }
        }
    })
}

//3. 导出
export default {
    Store,
    install
}
```

```
// 上面代码写完可以实现点击同步增加和同步删除
<button @click="increment">同步增加Num</button><br>
<button @click="decrement">同步减少Num</button><br>
```

解决点击增加异步action报错

```
[Vue warn]: Error in v-on handler: "TypeError: this.$store.dispatch is not a
function"
```

store/qlvuex.js

```
// 实现qlvuex


let Vue

//2. 这里Store是用于做数据存储的
class Store {
    //13. 这里需要看Store是在哪里new的，是在index.js里面new的时候传入了一个option配置项过
来
    // 这里接收所有的配置项
    constructor(options) {
        //14. 如何接收，这里可以定义一个属性，可以拿到options里面的值，但是最好不要这样写
        // this._s = options.state;
        //15. 因为这个数据需要是响应式的，发生改变之后需要通知页面更新，那么这个数据必须是响应
式的
        //16. 如何变成响应式的数据呢，只需要添加进new Vue中即可
        this.vm = new Vue({
            data: {
                state: options.state
            }
        })
        //18.如何解决getNum报错呢？ for getters
        // 在index.js中如果有 getters那么直接拿过来用，如果别人没有写那么设置默认值
        //19. 上面这个是取得是配置项中的getters
        let getters = options.getters || {}
        //20.下面是取的是上面store里面的一个空的属性，因为我们到时候访问是通过
```

```
        // this.$store.getters.getNum 访问的，那么肯定就是访问下面的getters
        this.getters = {}
        //21. 把getters中属性定义到this.getters，因为getters里面可能有多个方法，不只getNum一个
        // 那么就需要找到每一个key  getterName是每一个方法的名字
        Object.keys(getters).forEach(getterName => {
            //22. 定义一个属性，把每一个getter(比如getNum)，定义到this.getters ={} 对象中来
            Object.defineProperty(this.getters, getterName, {
                //23. 只有get方法，访问getterName属性就会触发get方法
                get: () => {
                    //24. getters 是对象，getName是函数名，比如这里就是getters[getNum]()取到getNum函数
                    // 并且传入参数 this.state
                    return getters[getterName](this.state)
                }
            })
        })
        // 24. 解决点击事件函数错误 for mutations
        let mutations = options.mutations || {}
        this.mutations = {}
        Object.keys(mutations).forEach(mutationName => {
            this.mutations[mutationName] = payload => {
                // 根据这个mutations[mutationName] 可以找到具体的函数
                mutations[mutationName](this.state, payload);
            }
        })
        //26. for actions
        let actions = options.actions || {}
        this.actions = {}
        Object.keys(actions).forEach(actionName => {
            this.actions[actionName] = payload => {
                // 这里的this就是 Store对象
                actions[actionName](this, payload)
            }
        })
    }
    //27. dispatch
    dispatch (type, payload) {
        this.actions[type](payload)
    }
    // 27步完之后还有报错Uncaught TypeError: Cannot read properties of undefined (reading 'mutations')
    // 28 因为代码在setTimeout里面调用就出现问题了
    //25. 上面写完点击会报错 vue.runtime.esm.js?c320:4603 [Vue warn]: Error in v-on handler: "TypeError: this.$store.commit is not a function"
    // 没有commit方法，commit传入了 参数的名字和payload
    // commit (type, payload) {
    //  // 这里面已经有方法了
    //  this.mutations[type](payload)
    // }
    // 28.那么我们就需要将上面的代码改成箭头函数来解决这个问题了
    commit = (type, payload) => {
        // 这里面已经有方法了
        this.mutations[type](payload)
    }
    //17. 这个数据如何访问呢？添加一个get方法，那么页面上第一种方式获取数据就可以显示了
    get state () {
```

```
        // 这里返回state对象，那么就可以使用this.$store.state.num了
        return this.vm.state
    }
}

//1. 在vue里面使用插件，一定要有这样一个方法，用于安装插件
// 安装插件目的是让每个组件都有一个 $store 1~12
// 3. 这里默认有一个参数 安装插件的时候会把Vue实例传过来
const install = (_Vue) => {
    Vue = _Vue;
    //4.这里有个mixin方法 给每一个组件都注册一个方法，beforeCreate
    Vue.mixin({
        //5. 这里写了之后意味着每个组件都有beforeCreate这样生命周期的一个方法
        beforeCreate () {
            //6. 每个组件注册的时候都会有个日志
            // console.log(this)
            //7. 如果不容易找到那么就在main.js里面加个名字 root 方便判断是那个节点先出来
            // console.log(this.$options)
            //8. 先出来根节点root 后app
            console.log(this.$options.name)
            //9. 如果说这个组件存在，并且上面有store，在main.js里面传入了一个store，
store的入口是在根组件上
            if (this.$options && this.$options.store) {
                //10. 那么就可以说明这个一定是根节点
                this.$store = this.$options.store
            } else {
                //11. 子组件  如果父组件存在，并且父组件中也存在 store
                this.$store = this.$parent && this.$parent.$store
                //12. 到这一步 store的报错就解决了，那么还剩下一个num的错误
                //Error in render: "TypeError: Cannot read properties of
undefined (reading 'num')"
            }
        }
    })
}

//3. 导出
export default {
    Store,
    install
}
```

所有代码整理

main.js

```
import Vue from 'vue'
import App from './App.vue'
import store from './store'

Vue.config.productionTip = false

new Vue({
  // 7. 添加名字方便查看
  name: "root",
  store,
  render: h => h(App)
}).$mount('#app')
```

store/index.js

```
import Vue from 'vue'
// import Vuex from 'vuex'  // 引用第三方库
import Vuex from './qlvuex' //5. 使用自己实现的qlvuex
//5.1 注释之后发现页面报错没有找到$store，那么第一步给每个组件都添加一个$store
// 顺序：先给根对象赋$store然后给子赋值的时候，判断父有没有，如果有就拿到子中 见
ql_vuex.drawio


Vue.use(Vuex)  // 使用插件，插件插入vue里面才能使用

// 每一个vue实例中都有一个属性$store
export default new Vuex.Store({
  state: {
    // 存储数据，怎么将num展示在页面上呢？
    num: 1
  },
  //2.2 方式二：添加一个getter，在这里面可以添加一个方法
  getters: {
    // 只读，并且可以对数据进行处理 参数是固定的
    getNum (state) {
      return state.num * 2
    }
  },
  mutations: {
    //3. mutations 相当于是同步的方式 参数是固定的
    // payload 是传入的参数
    //3.1 同步增加
    syncIncrement (state, payload) {
      state.num += payload
    },
    //3.1 同步减少
    syncDecrement (state, payload) {
      state.num -= payload
    }
  },
  actions: {
    //4. actions 相当于是异步的方式
    //4.1 异步增加
    asyncIncrement ({ commit, dispatch }, payload) {
      // 模拟ajax
      setTimeout(() => {
```

```
        commit("syncIncrement", payload)
      }, 1000)
    }
  },
  modules: {
  }
})
```

store/qlvuex.js

```
// 实现qlvuex


let Vue

//2. 这里Store是用于做数据存储的
class Store {
    //13. 这里需要看Store是在哪里new的，是在index.js里面new的时候传入了一个option配置项过来
    // 这里接收所有的配置项
    constructor(options) {
        //14. 如何接收，这里可以定义一个属性，可以拿到options里面的值，但是最好不要这样写
        // this._s = options.state;
        //15. 因为这个数据需要是响应式的，发生改变之后需要通知页面更新，那么这个数据必须是响应式的
        //16. 如何变成响应式的数据呢，只需要添加进new Vue中即可
        this.vm = new Vue({
            data: {
                state: options.state
            }
        })
        //18.如何解决getNum报错呢？ for getters
        // 在index.js中如果有 getters那么直接拿过来用，如果别人没有写那么设置默认值
        //19. 上面这个是取得是配置项中的getters
        let getters = options.getters || {}
        //20.下面是取的是上面store里面的一个空的属性，因为我们到时候访问是通过
        // this.$store.getters.getNum 访问的，那么肯定就是访问下面的getters
        this.getters = {}
        //21. 把getters中属性定义到this.getters，因为getters里面可能有多个方法，不只getNum一个
        // 那么就需要找到每一个key  getterName是每一个方法的名字
        Object.keys(getters).forEach(getterName => {
            //22. 定义一个属性，把每一个getter(比如getNum)，定义到this.getters ={} 对象中来
            Object.defineProperty(this.getters, getterName, {
                //23. 只有get方法，访问getterName属性就会触发get方法
                get: () => {
                    //24. getters 是对象，getName是函数名，比如这里就是getters[getNum]()取到getNum函数
                    // 并且传入参数 this.state
                    return getters[getterName](this.state)
                }
            })
        })
        // 24. 解决点击事件函数错误 for mutations
        let mutations = options.mutations || {}
        this.mutations = {}
```

```
        Object.keys(mutations).forEach(mutationName => {
            this.mutations[mutationName] = payload => {
                // 根据这个mutations[mutationName] 可以找到具体的函数
                mutations[mutationName](this.state, payload);
            }
        })
        //26. for actions
        let actions = options.actions || {}
        this.actions = {}
        Object.keys(actions).forEach(actionName => {
            this.actions[actionName] = payload => {
                // 这里的this就是 Store对象
                actions[actionName](this, payload)
            }
        })
    }
    //27. dispatch
    dispatch (type, payload) {
        this.actions[type](payload)
    }
    // 27步完之后还有报错Uncaught TypeError: Cannot read properties of undefined
(reading 'mutations')
    // 28 因为代码在setTimeout里面调用就出现问题了
    //25. 上面写完点击会报错 vue.runtime.esm.js?c320:4603 [Vue warn]: Error in v-on
handler: "TypeError: this.$store.commit is not a function"
    // 没有commit方法，commit传入了 参数的名字和payload
    // commit (type, payload) {
    //  // 这里面已经有方法了
    //  this.mutations[type](payload)
    // }
    // 28.那么我们就需要将上面的代码改成箭头函数来解决这个问题了
    commit = (type, payload) => {
        // 这里面已经有方法了
        this.mutations[type](payload)
    }
    //17. 这个数据如何访问呢？添加一个get方法，那么页面上第一种方式获取数据就可以显示了
    get state () {
        // 这里返回state对象，那么就可以使用this.$store.state.num了
        return this.vm.state
    }
}

//1. 在vue里面使用插件，一定要有这样一个方法，用于安装插件
// 安装插件目的是让每个组件都有一个 $store 1~12
// 3. 这里默认有一个参数 安装插件的时候会把Vue实例传过来
const install = (_Vue) => {
    Vue = _Vue;
    //4.这里有个mixin方法 给每一个组件都注册一个方法，beforeCreate
    Vue.mixin({
        //5. 这里写了之后意味着每个组件都有beforeCreate这样生命周期的一个方法
        beforeCreate () {
            //6. 每个组件注册的时候都会有个日志
            // console.log(this)
            //7. 如果不容易找到那么就在main.js里面加个名字 root 方便判断是那个节点先出来
            // console.log(this.$options)
            //8. 先出来根节点root 后app
            console.log(this.$options.name)
```

```
            //9. 如果说这个组件存在，并且上面有store，在main.js里面传入了一个store，
store的入口是在根组件上
            if (this.$options && this.$options.store) {
                //10. 那么就可以说明这个一定是根节点
                this.$store = this.$options.store
            } else {
                //11. 子组件  如果父组件存在，并且父组件中也存在 store
                this.$store = this.$parent && this.$parent.$store
                //12. 到这一步 store的报错就解决了，那么还剩下一个num的错误
                //Error in render: "TypeError: Cannot read properties of
undefined (reading 'num')"
            }
        }
    })
}

//3. 导出
export default {
    Store,
    install
}
```