# vue

## 上下级组件之间改变状态

在1.x版本上的:propertyName.sync传参到子组件，当子组件的属性值改变时，父组件也会变，在2.0版本废除后，2.3版本之后又恢复了sync；所以使用 :propertyName.sync，绑定属性时的属性值sync即可；

父组件使用：

<select-input :is-show.sync="listShow"></select-input>

<list v-show="listShow"></list>

select-input子组件：

<input @click="showListHandle" />

<script>

export default {

props:['isShow'],

computed:{

initShow(){

return this.isShow

}

},

methods: {

showListHandle(){

//触发更新，名字是is-show，值为initshow取反；

this.$emit("update:isShow", !this.initShow)

}

}

}

</script>

## 兄弟组件之间改变状态

需要使用父组件作为通道，父组件传入子组件自定义的事件，在子组件触发某个事件时，使用$emit来触发父组件传入的自定义事件，来改变兄弟组件的状态；

需求：

父组件的title值传入了select-input组件作为input的值；

list组件的列表项点击时需要改变title值；

以改变兄弟组件select-input的input值；

做法是：

父组件传入list组件自定义事件changeTitle；

在list组件的列表项点击时，触发getTitleHandle事件处理函数，然后再来触发自定义事件changeTitle来改变父组件的title值；

父组件

<select-input :is-show.sync="listShow" :title=”title”></select-input>

<list v-show="listShow" @changeTitle’=”titleHandle”></list>

<script>

exprot default{

data(){

return {

listShow: flase;

title:’’

}

},

methods:{

titleHeadle(title){

this.title=title;

}

}

}

</script>

select-input组件

<input @click="showListHandle" :value="this.props.title"/>

list组件：

<template>

<ul class="list">

<li v-for="item in data" @click="getTitleHandle(item.title)">{{item.title}}</li>

</ul>

</template>

<script>

exprot default{

data(){

},

methods:{

getTitleHandle(title){

this.$emit(‘changeTitle’,title)

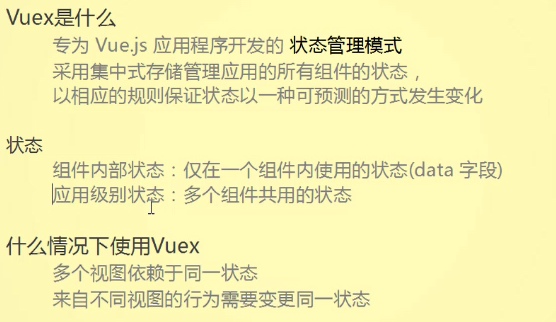
}

}

}

</script>

# 使用vuex



安装模块

cnpm install vuex --save

作为插件

Vue.use(Vuex)

定义容器

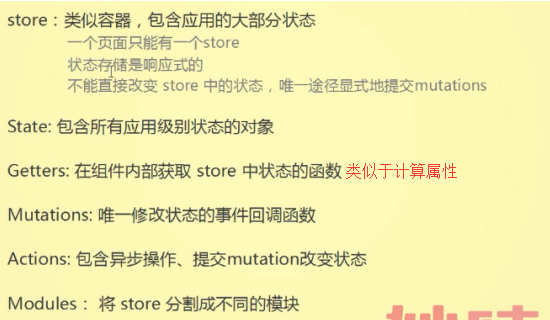
new Vuex.Store()

注入根实例

{

store

}



import Vue from 'vue'

import Vuex from 'vuex'

Vue.use(Vuex)

let store = new Vuex.Store({

state:{

count:100

},

getters:{

//默认传入第一个参数为store.state;

filter(state){

return state.count >= 120 ? 120 : state.count;

}

},

mutaitons:{

//默认传入第一个参数为store.state;

add(state,n,m){

state.count+=n;

},

de(state,payload){

state.count-=payload.n;

},

test(state,payload){

state.count-=payload.de;

}

},

actions:{

//默认传入第一个参数当前store对象实例；

add(context){

setTimeout( ()=>{

//vuex中，改变状态必须提交mutations；

//所以这里的add是mutations的add方法

context.commit(‘add’, {n:5} )

}, 1000)

}

}

})

export default store

## 读取state的属性

this.$store.state.count;

## 读取getters的属性

this.$store.getters.filter;

## mutaitons对象的方法调用同步改变state的属性值

mutations对象改变state属性值时，都是同步的；

this.$store.commit(‘add’,5,3); //名字、参数

this.$store.commit(‘de’,{

n:5

}); //名字、参数对象

this.$store.commit({

type:’test’,

de:5

}); //对象形式

## actions对象的方法调用异步改变state的属性值

this.$store.dispatch(‘add’)

# vuex辅助函数

mapState

mapGetters

mapMutations

mapActions

以上方法都接收一个参数，返回对象；

import {mapState,mapGetters,mapActions,mapMutations} from 'vuex'

/\*多种写法

computed: mapState({

num: state => state.count,

num: 'count',

num(state) {

return state.count+100

},

count: 'count'

}),

computed: mapState(['count']),

\*/

export default {

computed: {

...mapGetters({

num2: 'filterCount'

}),

...mapState(['count']),

},

methods: {

...mapActions({

addHandle: 'addAction'

}),

...mapMutations({

deHandle:'deIncrement'

})

}

}

如需传参可以直接写在绑定事件上：

<input type="button" value="-" @click="deHandle({de:5})" />

# vuex模块

let selectModule = {

state:{

title:'hello123',

list: []

},

mutations:{

changeTitle(state, payload){

state.title = payload.title

},

changeList(state, list){

state.list = list;

}

},

actions:{

getListAction({commit}){

// 发送请求

axios.get('http://easy-mock.com/mock/594f5d4b9adc231f3569be76/list/list')

.then((data)=>{

commit("changeList", data.data); // 拿到数据后，提交mutations，改变状态

})

.catch((error)=>{

conso.log(error)

})

}

}

};

// 定义一个容器

let store = new Vuex.Store({

modules:{

selectModule

}

})

state里的属性需要如下访问，其他的不变；

// this.$store.state.selectModule.title

//this.$store.commit(‘changeTitle’,5,3);

//this.$store.dispatch(‘getListAction’,5,3);

# vuex流程图

组件触发事件dispatch；

actions调用后端接口；

调用commit，来触发mutations；

改变state的状态；

最后渲染组件；

