Vue全家桶入门和企业级项目开发

第2天课堂笔记

讲师：邵山欢（考拉老师）

# 一、日历项目细看一下

## 1.1 webpack4的事情

webpack4做出了两个比较大的改变：提出了webpack-cli、增加了mode属性。

**webpack4必须写mode这个属性**，有两个值：production（上线版）、development（开发版）。

**我们开发的时候必须要设置mode:development，要不然代码都被混淆加密，打包很慢**。

const path = require('path');

module.exports = {

entry: "./www/app/main",

output: {

path: path.resolve(\_\_dirname, "./www/dist"),

filename: "bundle.js",

publicPath: "/xuni/" //这是对webpack-dev-server的配置

},

module: {

rules: [

{

test: /\.vue$/,

include: [

path.resolve(\_\_dirname, "www/app")

],

exclude: [

path.resolve(\_\_dirname, "node\_modules")

],

loader: "vue-loader",

}

]

},

resolve: {

alias: {

'vue$': 'vue/dist/vue.esm.js' // 用 webpack 1 时需用 'vue/dist/vue.common.js'

}

},

mode: 'development'

}

建议大家都用最新版的工具，自动有热更新、打包很快：

npm install -g webpack@4

npm intssall -g webpack-dev-server@3

## 1.2 说一下:is

我们最常见的业务就是通过一个值来决定这里放哪个组件。

React中，用函数

render(){

const showPicker = () => {

switch(this.state.view){

case "year" :

return <YearView />

case "month" :

return <MonthView />

}

}

return <div>

{showPicker()}

</div>

}

Vue中这样用函数有问题，因为我们没有jsx，组件必须是v-html来生成。画风就不好：

<template>

<div>

<div **:is="'my-' + view + 'picker'"** :date="date" :setView="setView"></div>

<div **:is="'my-' + view + 'view'"** :date="date" :setView="setView"></div>

</div>

</template>

React中的生命周期是我们理解的，先进行componentWillReceiveProps()、再进行……、再进行render。

vue中因为底层是变量劫持，所以dom和数据绑的特别密切，数据一变化dom自动变化。但是为了也有生命周期的概念，vue的作者就在Object.defineProperties()中的setter中主动调用了生命周期的钩子。

# 二、vuex的配置和使用

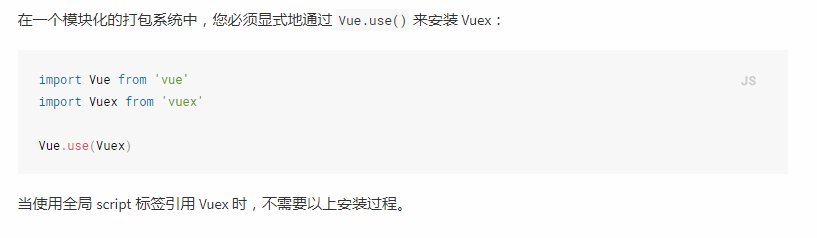
## 2.1 store、state、mutations、actions文件的基本写法

vuex类似于redux + dva的可预测状态的容器，非常类似dva，换句话说dva就是模仿vuex开发的。

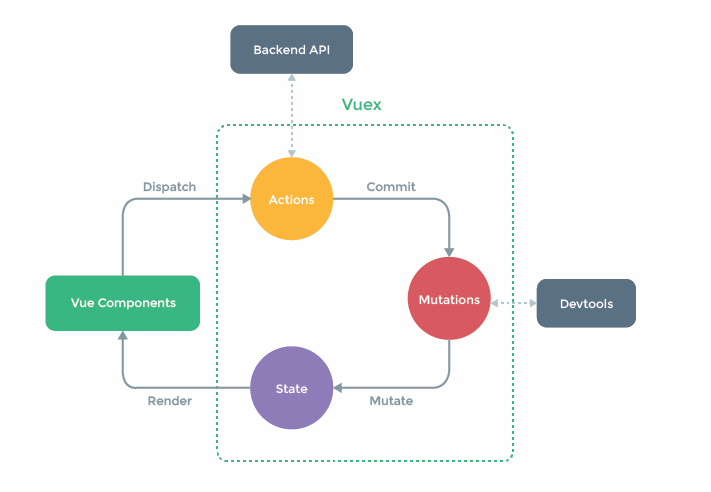
安装：

cnpm i --save vuex

如何使用：



昨天我们强调了：一个vue项目中只有一次new Vue()的机会。相当于我们new Vue()之前，这个Vue构造函数有机会被“装饰”。Vue.use()就是一个装饰Vue的过程，表示使用插件。



创建store文件夹，创建store.js文件，先写一个基本的样子：

import Vue from "vue";

**import Vuex from "vuex";**

**Vue.use(Vuex);**

**export default new Vuex.Store({**

**state: {**

**count: 0**

**},**

**//同步变化，没有任何副作用的变化**

**mutations: {**

**add(state) {**

**//注意不是纯函数，可以直接进行一个值的改变**

**state.count++**

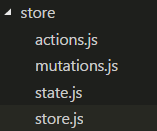
**}**

**}**

**});**

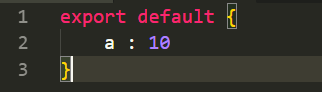
vuex官网的一些案例：https://github.com/vuejs/vuex/tree/dev/examples

这个文件一般都是拆分出来的：



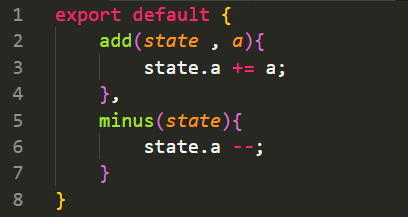
下面我们看每个文件的画风：

state.js:



这里就是暴露一个数据，不一定是对象，可以直接暴露数组。就是redux中的state。

mutations.js:



**这里是唯一可以改变state的地方，没有其他任何地方能够改变state了**！！

mutaions.js文件需要注意的是：

1）mutations类比dva中的reducers，但是不是纯函数了，它每一个函数都要直接改变state的值。为什么不需要纯函数了呢？因为不需要进行diff算法比较了，vuex的视图更新策略也是变量劫持，而不是所谓的给组件传入一个props，不是Provider、connect那一套。

2）没有载荷的概念了，所有的参数都是通过“参数”的形式，罗列在括号中。

actions.js:



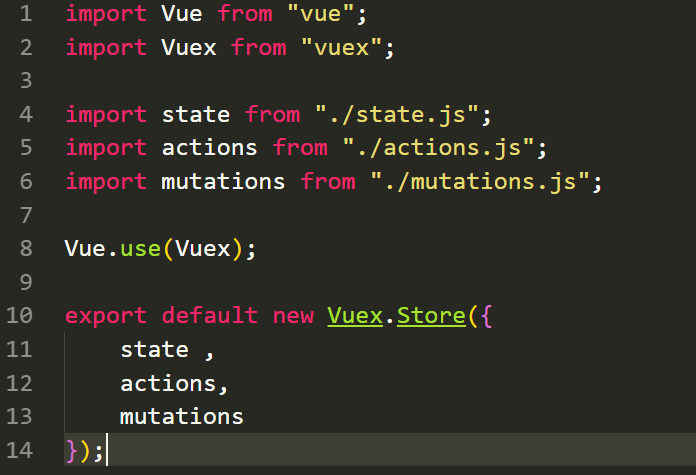
**actions.js文件是写有副作用的改变的地方，类比于我们dva中的effects**，比如异步、if语句等等都叫做副作用。副作用结束，使用commit()来调用一个同步的mutation。即，**这里面罗列的所有函数，都必须有commit()语句，否则没有任何意义**。

actions.js文件需要注意：

1）没有JSON类型的action的概念了，action就是一个字符串的名字加上罗列的参数，而不是redux中的{type : "add" , "a" : 2}的形式了，而是commit("add" , 2);

2）这里面要复现一下mutations.js中的所有函数，可以和mutaions.js中的函数同名，换句话说，必须同名，看的清除。并且这些函数一般罗列在前几名。为什么要罗列一下呢？因为组件中有一个mapActions的函数，只能罗列action不能罗列mutation。

store.js:



注意要写export default默认暴露，罗列state、actions、mutations。

这个文件要注意，这个文件不能拆！！要拆拆state、actions、mutations去，一会儿我们做演示。

app.vue组件文件：



**组件中使用全局的值，在template插值中写{{$store.state.a}}，在methods、computed的函数中写this.$store.state.a即可**。

尽量使用vuex提供的mapActions函数，这个函数返回一个对象，对象就直接和全局的store进行了映射。

**如果不映射，我们就要手动写this.$store.commit()或者this.$store.dispatch()，见下图**。你别redux**陋习**，还this.props.dispatch()呢。



## 2.2 异步数据的请求

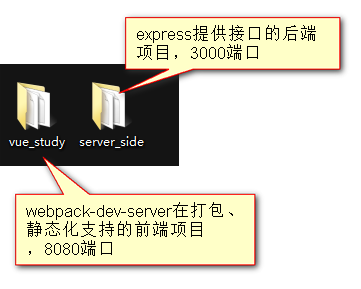
### 2.2.1 关于后端项目新文件

我们在react课上的套路是后端的数据接口/api也是跑在和前端同一个接口下。前端项目是用node app.js跑起来的，www文件夹被静态了。即请求的发出组件和/api接口都是同一个端口，不涉及跨域问题。

但是，工作中，一定是要跨域请求数据的。

**工作中，每个人的电脑都是同一个局域网的，你的后台哥哥会给你一个后端接口文档，接口在演示的时候，很有可能是3000、24333等等。你的webpack-dev-server是8080端口。**

创建一个后端的文件夹，这是一个独立的文件夹，



app.js:

var express = require("express");

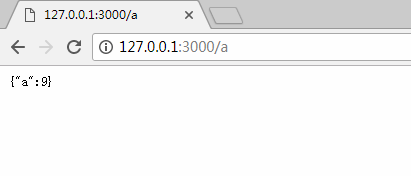
var app = express();

app.get("/a" , (req,res)=>{

res.json({"a" : 9})

});

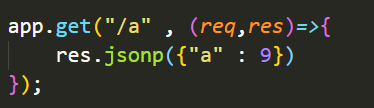
app.listen(3000);

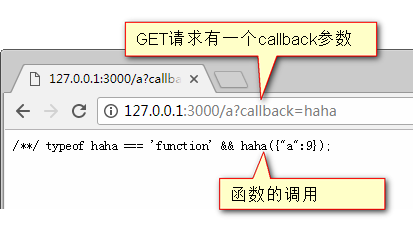


### 2.2.2 关于跨域-方法1 JSONP

● 方法1： JSONP，后端给你的是一个函数的调用，而不是数据。数据通过实参传入页面。

后端提供jsonp接口：





此时前端就用$.ajax()进行跨域：





### 2.2.3 关于跨域-方法2 CROS

● 方法2：CROS跨域

禁止8080的前端页面请求3000的后端页面，是谁禁止的？是浏览器。此时我们就要写一个HTTP下行响应的头部，来告诉浏览器，这次请求是允许的。

要想面试回答问题回答的好，读这个文章：

**https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Access\_control\_CORS**





所以我们要改变nodejs程序，让下行的响应加上

Access-Control-Allow-Origin: \*

直接看express的手册，写一个头部：

var express = require("express");

var app = express();

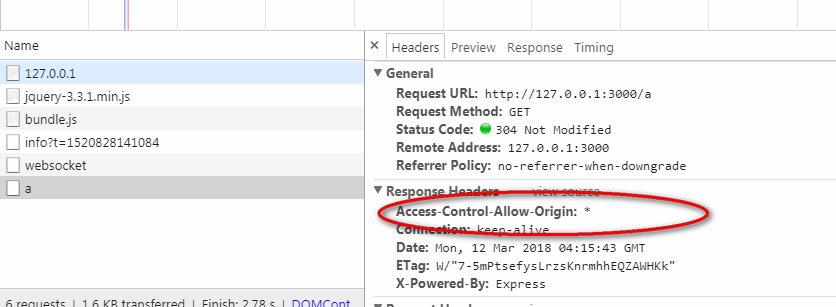
app.get("/a" , (req,res)=>{

**res.setHeader("Access-Control-Allow-Origin" , "\*");**

res.json({"a" : 9})

});

app.listen(3000);



### 2.2.4 关于跨域-方法3 proxy

● 方法3：代理跨域

让http://127.0.0.1:**3000**/a这个接口，被“代理”到http://127.0.0.1:**8080**/api/a

相当于服务器去“偷数据”，由后端（前端的后端，就是webpack-dev-server出面），问3000要数据。

我们拿PHP做例子：

<?php

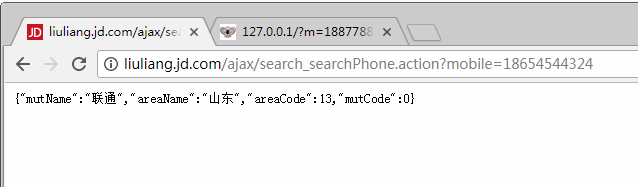
echo file\_get\_contents("http://www.baidu.com");

?>

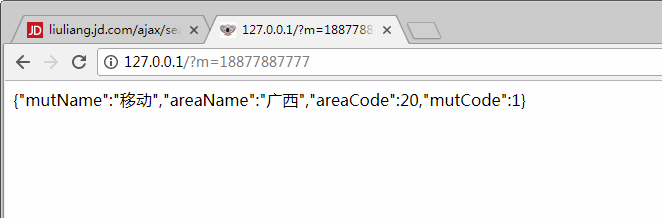
此时我们的80端口一运行，就是百度！！



比如京东：



可以被后端偷来：



webpack-dev-server也可以通过简单的配置进行后代的“代理”跨域：

https://doc.webpack-china.org/configuration/dev-server/#devserver-proxy

webpack-dev-server这个小型服务器底层也是express。

也就是说，webpack-dev-server现在在给我们做三件事儿：

**1）构建app文件夹到/xuni/bundle.js去**

**2）静态化www文件夹提供了静态文件的路由**

**3）代理了127.0.0.1:3000的接口。**

**4）构建app的时候vue那些东西进行了翻译**

这三件事儿，靠webpack.config.js配置的：

const path = require('path');

module.exports = {

**entry: "./www/app/main",**

**output: {**

**path: path.resolve(\_\_dirname, "./www/dist"),**

**filename: "bundle.js",**

**publicPath: "/xuni/" //这是对webpack-dev-server的配置**

**},**

**module: {**

**rules: [**

**{**

**test: /\.vue$/,**

**include: [**

**path.resolve(\_\_dirname, "www/app")**

**],**

**exclude: [**

**path.resolve(\_\_dirname, "node\_modules")**

**],**

**loader: "vue-loader",**

**}**

**]**

**},**

mode: 'development',

**devServer : {**

**proxy: {**

**"/api": {**

**target: 'http://www.iqianduan.cn',**

**changeOrigin: true,**

**pathRewrite: {**

**'^/api': '/'**

**}**

**}**

**},**

**contentBase: path.resolve(\_\_dirname, "./www"),**

**port : 8080**

**},**

resolve: {

alias: {

'vue$': 'vue/dist/vue.esm.js' // 用 webpack 1 时需用 'vue/dist/vue.common.js'

}

}

}

总结一下：

一提“跨域”，立马等回答他一共三种方法。

一提“Promise”，要有套路回答。

# 三、做一下Vuex版本的TodoList

服务器端用变量来持久，即重新node app.js数据丢失，不用mongodb和fs模块了。

因为毕竟是学习前端的。

## 3.1 后端RESTful风格接口的书写

var todos = [

{"id" : 1 , "title" : "吃饭" , "done" : false}

];

//列出所有todos：

app.get("/todos" , (req,res)=>{

res.json({todos});

});

//增加todos:

app.post("/todos" , (req,res)=>{

var form = new formidable.IncomingForm();

form.parse(req, (err , {title})=>{

//寻找最大的id

var id = todos.reduce((a,b)=>{b.id > a ? b.id : a} , 0) + 1;

todos.push({

id,

title,

done : false

});

res.json({ todos });

});

});

//删除todo

app.delete("/todos/:id" , (req,res)=>{

todos = todos.filter(item=>item.id != req.params.id);

res.json({ todos });

});

//改变todo

app.patch("/todos/:id" , (req,res)=>{

var form = new formidable.IncomingForm();

form.parse(req, (err, { k , v }) => {

//寻找最大的id

todos = todos.map(item => item.id == req.params.id ? {...item , [k] : v} : item);

res.json({ todos });

});

});

## 3.2 说一下vuex的logger

import Vue from "vue";

import Vuex from "vuex";

**import createLogger from 'vuex/dist/logger';**

import state from "./state.js";

import actions from "./actions.js";

import mutations from "./mutations.js";

Vue.use(Vuex);

export default new Vuex.Store({

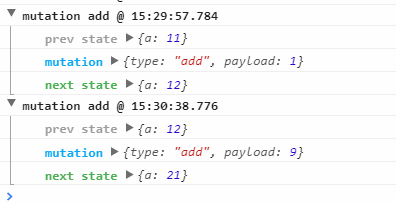
state ,

actions,

mutations,

**plugins: [createLogger()]**

});



## 3.3 拆分actions、mutations

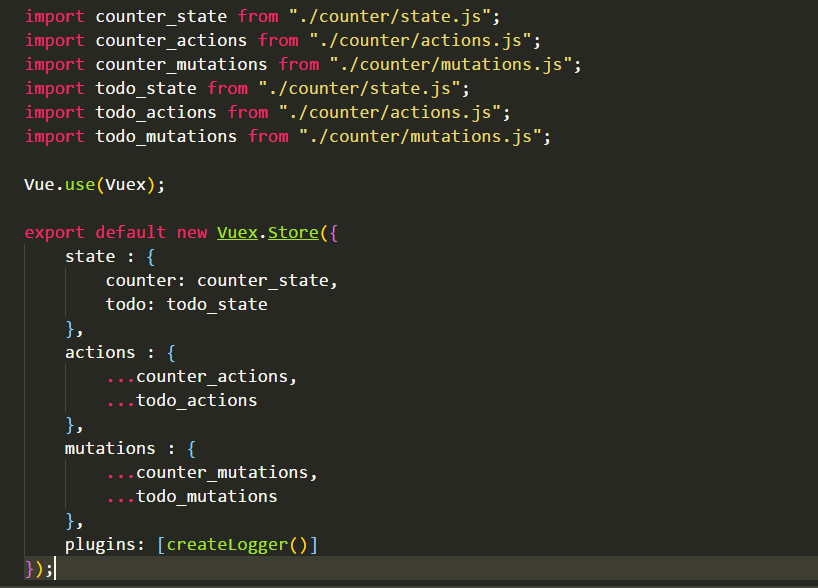
注意store不能拆分的，也就是说不是dva那种

app.model(counter)

app.model(todo)

而是redux那种感觉

combindReducers()



vue的组件不分通不通天，所有组件都沐浴在了$store下，也都有能力this.$store.dispatch()、this.$store.commit()东西。

**所以，我们react课上说，map的东西不要通天，调用父亲的函数，在vue中没有这个规定。**

## 3.4 开发

开发没什么笔记好写，写一些重点。

● 在created()构造器中发init。

● 复选框要受控，从$store中要值，改变能改变store的值，且发出patch请求，此时不要用v-model：

<template>

<div class="bar">

<input type="checkbox" v-model="done" />

{{item.title}}

<button @click="deltodo">删除</button>

</div>

</template>

用v-model就势必会watch：

<script>

import Vue from "vue";

export default {

props : ["item"],

data(){

return {

done : this.item.done

}

},

methods : {

deltodo(){

this.$store.dispatch("deltodo" , {"id" : this.item.id});

}

},

watch : {

done(){

alert("你试图改变done")

}

}

};

</script>

而应该直接用:checked="" 和@click。

注意**复选框的onclick事件和onchange事件是同一个事件！！不是vue的规定，是w3c的规定。**

## 3.5 新知识点：getters

在vuex中，getters类似组件中的computed，表示state的一些计算后的值。

**比如todos中有2个计算后的值：1）所有未做 2）所有已做**

**在react中，筛选todo是在组件中完成的，画风**：

render(){

const arr = (()=>{

if(**this.props.showtype == "SHOWALL"**){

return this.props.todos;

}else if(**this.props.showtype == "ONLYSHOWDONE"**){

return this.props.todos.**filter(item=>item.done);**

}else if(**this.props.showtype == "ONLYSHOWUNDONE"**){

return this.props.todos.**filter(item=>!item.done);**

}

})();

return <div>

{

arr.map(item=>{

return <li></li>

})

}

</div>

}

在vue中提出了getters的概念，此时我们将这样的过滤、筛选、有关数据、计算后的数据都放在getters中。

创建getters.js文件：

export default {

undonetodos(state){

return state.todo.todos.filter(item=>!item.done);

},

donetodos(state) {

return state.todo.todos.filter(item => item.done);

}

}

地道的vue开发者，

组件方法1：

<todo\_bar

v-if="$store.state.todo.showtype == 'SHOWALL'"

v-for="item in todos" :key="item.id" :item="item"

></todo\_bar>

<todo\_bar

v-if="$store.state.todo.showtype == 'ONLYSHOWUNDONE'"

v-for="item in $store.getters.undonetodos" :key="item.id" :item="item"

></todo\_bar>

<todo\_bar

v-if="$store.state.todo.showtype == 'ONLYSHOWDONE'"

v-for="item in $store.getters.donetodos" :key="item.id" :item="item"

></todo\_bar>

方法2：

<template>

<div class="bd">

<todo\_bar

v-for="item in todos" :key="item.id" :item="item"

></todo\_bar>

</div>

</template>

<script>

import Vue from "vue";

import todo\_bar from "./todo\_bar.vue";

export default {

components : {

"todo\_bar" : todo\_bar

},

computed : {

todos(){

if(this.$store.state.todo.showtype == "SHOWALL"){

return this.$store.getters.alltodos;

}else if(this.$store.state.todo.showtype == "ONLYSHOWDONE"){

return this.$store.getters.donetodos;

}else if(this.$store.state.todo.showtype == "ONLYSHOWUNDONE"){

return this.$store.getters.undonetodos;

}

}

}

};

</script>

<style>

.bd{

min-height: 255px;

}

</style>

# 四、vue的小伙伴们

## 4.1 less和stylus

<style **lang="less"**>

.cur{

background-color: gold;

}

</style>

此时就要去vue-loader中查询less如何配置。

https://vue-loader.vuejs.org/



需要改变webpack.config.js文件的module部分：

module: {

rules: [

{

test: /\.vue$/,

include: [

path.resolve(\_\_dirname, "www/app")

],

exclude: [

path.resolve(\_\_dirname, "node\_modules")

],

loader: "vue-loader",

**options: {**

**loaders: {**

**less: 'vue-style-loader!css-loader!less-loader'**

**}**

**}**

}

]

},

## 4.2 iview ui框架

如同antd，vue中也有一些ui框架，比较有名的：elemeui、iviewui。

https://www.iviewui.com/

安装依赖：

cnpm install --save iview

引用样式表（注意css要从node\_modules中copy出来）：

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Document</title>

**<link rel="stylesheet" href="css/iview.css">**

</head>

<body>

<div id="app"></div>

<script src="/jslib/jquery-3.3.1.min.js"></script>

<script src="/xuni/bundle.js"></script>

</body>

</html>

配置很简单，入口文件main.js:

main.js:

import Vue from "vue";

import Vuex from "vuex";

import iview from "iview";

import storeobj from "./store/store.js";

import app from "./components/app.vue";

Vue.use(iview);

Vue.use(Vuex);

const store = new Vuex.Store(storeobj);

new Vue({

el : "#app" ,

store,

render : h => h(app)

});

组件里面无脑使用组件，组件不需要注册。

<template>

<div>

<Button>Default</Button>

<DatePicker type="date" placeholder="Select date" style="width: 200px"></DatePicker>

<Rate v-model="fenshu"></Rate> {{fenshu}}

<Dropdown>

<a href="javascript:void(0)">

下拉菜单

<Icon type="arrow-down-b"></Icon>

</a>

<DropdownMenu slot="list">

<DropdownItem>驴打滚</DropdownItem>

<DropdownItem>炸酱面</DropdownItem>

<DropdownItem disabled>豆汁儿</DropdownItem>

<DropdownItem>冰糖葫芦</DropdownItem>

<DropdownItem divided>北京烤鸭</DropdownItem>

</DropdownMenu>

</Dropdown>

</div>

</template>

不管是elemeui还是iviewui，几乎所有组件都能够直接v-model。

## 4.3 路由

路由：https://router.vuejs.org/zh-cn/

npm install vue-router

我们改变入口文件：

import Vue from "vue";

import Vuex from "vuex";

import iview from "iview";

**import VueRouter from 'vue-router'**

import storeobj from "./store/store.js";

**import router from "./router.js";**

import app from "./components/app.vue";

Vue.use(iview);

Vue.use(Vuex);

**Vue.use(VueRouter);**

const store = new Vuex.Store(storeobj);

new Vue({

el : "#app" ,

store,

**router,**

render : h => h(app)

});

创建一个router.js文件：

import VueRouter from 'vue-router';

import xinwen from "./components/xinwen.vue";

import yinyue from "./components/yinyue.vue";

export default new VueRouter({

routes: [

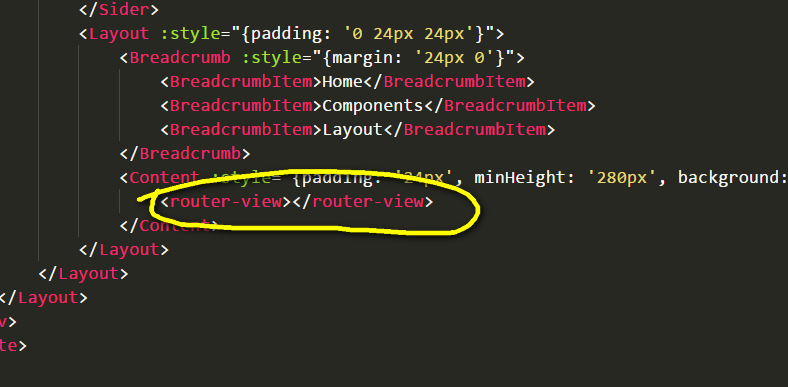
**{ path: '/xinwen', component: xinwen },**

**{ path: '/yinyue', component: yinyue }**

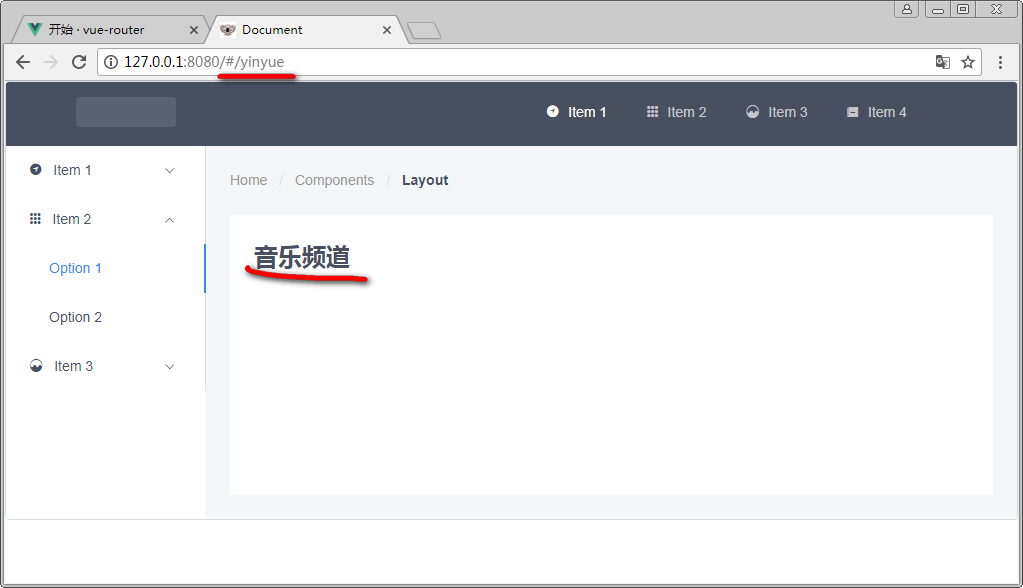
]

});

组件中放一个router-view组件：



此时：



超级链接：

<router-link to="/yinyue">音乐频道</router-link>

顶部header：

<Header>

<Menu mode="horizontal" theme="dark" :active-name="$router.history.current.fullPath">

<div class="layout-logo"></div>

<div class="layout-nav">

<MenuItem name="/yinyue">

<Icon type="ios-navigate"></Icon>

**<span @click="$router.push('/yinyue')">音乐频道</span>**

</MenuItem>

<MenuItem name="/xinwen">

<Icon type="ios-keypad"></Icon>

**<span @click="$router.push('/xinwen')">新闻频道</span>**

</MenuItem>

</div>

</Menu>

</Header>

音乐频道的子栏目：古典音乐、现代音乐、鬼畜音乐、嘻哈音乐

新闻频道的子栏目：中国新闻、国际新闻（亚洲新闻、非洲新闻、南极新闻）

路由是二分法，组件中只能有一个router-view标签。router-view就是小电视，靠路由表决定显示什么。

任何<router-view></router-view>中呈递什么组件，都要由路由表决定。

import VueRouter from 'vue-router';

import xinwen from "./components/xinwen/xinwen.vue";

import yinyue from "./components/yinyue/yinyue.vue";

import gudian from "./components/yinyue/gudian.vue";

import xiha from "./components/yinyue/xiha.vue";

import xiandai from "./components/yinyue/xiandai.vue";

export default new VueRouter({

routes: [

{

path: '/xinwen',

component: xinwen

},

{

path: '/yinyue',

component: yinyue ,

**children : [**

**{**

**path: 'gudian',**

**component: gudian**

**},**

**{**

**path: 'xiha',**

**component: xiha**

**},**

**{**

**path: 'xiandai',**

**component: xiandai**

**}**

**]**

}

]

});

yinyue.vue:

<template>

<div>

<Layout>

<Sider hide-trigger :style="{background: '#fff'}">

<Menu active-name="2-1" theme="light" width="auto" :open-names="['2']">

<MenuItem name="1-1">古典音乐</MenuItem>

<MenuItem name="1-2">现代音乐</MenuItem>

<MenuItem name="1-3">嘻哈音乐</MenuItem>

</Menu>

</Sider>

<Layout :style="{padding: '0 24px 24px'}">

<Breadcrumb :style="{margin: '24px 0'}">

<BreadcrumbItem>Home</BreadcrumbItem>

<BreadcrumbItem>Components</BreadcrumbItem>

<BreadcrumbItem>Layout</BreadcrumbItem>

</Breadcrumb>

<Content :style="{padding: '24px', minHeight: '280px', background: '#fff'}">

<h1>音乐频道</h1>

**<router-view></router-view>**

</Content>

</Layout>

</Layout>

</div>

</template>

<script>

import Vue from "vue";

export default {

}

</script>

<style lang="less">

</style>

有router-view必有一个children

有children必有一个router-view