vue-router

1安装（1）在模块化的打包系统中，须显示地通过Vue.use()来安装VueRouter

import Vue from ”vue”

import VueRouter from ”vue-router”

Vue.use(VueRouter)

（2）当使用全局script标签引用VueRouter时，不需要以上安装过程

2

html文件中有

<router-link to=”路由路径名1”></router-link>

<router-link to=”路由路径名2”></router-link>

…

<router-view></router-view>

javascript文件中有

const routers = [

{path:”路由路径名1”,component:组件名1},

{path:”路由路径名2”,component:组件名2}

]

const router = new VueRouter({

routers

})

vue实例中有router

3 动态路由匹配

1. route是{path:”/正常路径/:动态路径参数名1

/正常路径/:动态路径参数名2”,

component:符合以上路径后统一对应的组件名}

可使用组件实例.$route.params.动态路径参数名1

组件实例.$route.params.动态路径参数名2

1. 响应路由参数的变化：符合以上路径后统一对应的组件实例会被复用，即组件的生命周期钩子不会再被调用。复用组件时，想对路由参数的变化作出响应，可：

route的组件中有watch:{

”$route”(to,from){

}

}

1. 高级匹配模式：vue-router使用path-to-regexp作为路径匹配引擎，所以支持很多高级的匹配模式
2. 匹配优先级：有时同一个路径可以匹配多个路由，此时匹配的优先级就按照路由的定义顺序：谁先定义的，谁优先级越高

4 嵌套路由

route的组件的template中有<router-view></router-view>

route中有children:[

{path:”子路径1”,

component:符合routes的path+子路径1对应的组件名}，

{path:”子路径2”,

component:符合routes的path+子路径2对应的组件名}，

]

5 编程式的导航：效仿window.history API

1. <router-link :to=””> = router.push()

router.push(”user”)

router.push({path:”user”})

router.push({path:”user”,query:{username:”mjx”}})

router.push({name:”user”,params:{userId:123}})

1. <router-link :to=”” replace> = router.replace()
2. router.go(n) 在history记录中向前（正）或后（负）多少步

6 命名路由

route是{name:”user”,path:”/user/:userId”,component:User}

使用<router-link :to=”{name:’user’,params:{userId:123}}”>

</router-link>或

router.push({name:”user”,params:{userId:123}})

两种方式把路由导航到user/123路径

7 命名视图

有<router-view></router-view> 若没有name，默认为default

<router-view name=”a”></router-view>

<router-view name=”b”></router-view>

route是

{…

components:{

default:组件名1,

a:组件名2,

b:组件名3

}

}

8 重定向和别名

1. 重定向：当用户访问/a时，url将会被替换成/b，然后匹配路由为/b

route是{path:”/a”,redirect:”/b”}或

route是{path:”/a”,redirect:{name:”路径对象名”}} 或

route是{path:”/a”,redirect:to=>{

//接收目标路由为参数

//return 重定向的字符串路径或路径对象

}}

1. 别名：若/a的别名是/b则当用户访问/b时，url会保持为/b但路由匹配为/a，就像用户访问/a一样

route是{path:”/a”,alias:”/b”}

9 html5 history模式：vue-router默认has模式。可用：

const router = new VueRouter({

mode:”history”,

…

})

//修改

10 导航钩子

1. 全局钩子

const router = new VueRouter({…})

router.beforeEach((to,from,next)=>{})

当导航触发时，全局的before钩子按顺序调用，钩子是异步解析执行，此时导航在所有钩子resolve完之前一直处于等待中。to是即将进入的路由对象，from是正要离开的路由对象。next用于resolve这个钩子，必须有，next()会进行管道的下一个钩子next(false)中断当前导航，url重置from路由对象对应地址。next(”/”)或next({path:”/”})跳转到别的地址。

此处还有router.afterEach(route=>{})

1. 某个路由独享的钩子

route中有beforeEnter:(to,from,next)=>{}

1. 组件内的钩子

route的组件中有：

beforeRouteEnter(to,from,next){

//不能访问组件实例this

}

可通过：

beforeRouteEnter(to,from,next){

next(Vm=>{

//通过vm访问组件实例

})

}

有beforeRouteLeave(to,from,next){

//可以访问组件实例this

}

11路由元信息

1. route中有meta:{名:值}
2. 一个路由匹配到的所有路由记录会暴露为$route对象(或导航钩子中的route对象)的$route.matched数组，需遍历$route.matched来检查路由记录中的meta字段

12 过渡动效：<transition>的所有功能在这里同样适用

13 数据获取：可以在导航完成之前获取也可以在导航完成之后获取

14 滚动行为：只在html5 history模式下可用

const router = new VueRouter({

…

scrollBehavior(to,from,savedPosition){

//return

}

})

savedPosition当且仅当popstate导航（通过浏览器的前进/后退按钮触发）时才可用。

返回：滚动位置的对象信息如{x:number,y:number}或{selector:string}或savedPosition。若返回布尔假或空对象，则不会滚动

15 懒加载：用Vue的异步组件和webpack的code spliting feature 实现路由组件的懒加载