# 第七周总结

## 大致总结

### （一）本周学习内容：

1.Vue路由、Vuex。

2.复习javaScript语法、API,了解javaScript事件机制、父子窗口原理。

3.复习Java包装类(Integer，String,Boolean)、数据结构(collection，List,Set,Map,Queue)

4.复习数据库相关知识

### （二）遇到的问题及如何解决

1．在使用Vuex进行组件通信时，不明白为何actions中的commit可以直接调用到mutations中的函数，并且为何同属一个store？

解决：查看了Vuex的API,commit与actions、mutations一样，都是store对象下的属性,commit函数内部在实现对mutations属性的直接调用。

2.commit提交参数时，mutations接受不到参数？

解决：在传递组件参数时，commit提交的参数必须使用大括号**{}**包裹才能传递，并且mutations接收参数时也需要这样，mutaion(state,{prop})。

### （三）经验教训

为什么学过的东西总是容易忘，每次需要时总是花很多时间去看复习却收获甚少？

总结原因以改进有几点：

1. 对概念理解得不够深刻，一知半解，在真正使用时摸棱两可，不知道如何使用。

在熟悉相关知识点时，API固然重要，但是其最核心的应该是对其设计理念的理解。在实际项目中或许忘记了API的写法，但其API的概念、设计原理等却更加重要，会使得在开发中自然而然地将这些理念融入到你的项目当中去。

1. 没有及时对知识进行系统性的总结梳理，缺乏系统性的认识。

及时总结，及时回顾，做好笔记。

1. 在复习时最好一边看、一边实践，最好有一个demo

### （四）下周计划

1.做一个Vue的demo，学习Vue如何与后端交互

2.对java的基础知识继续跟进学习，包括IO、NIO、异常处理、多线程技术

## 学习笔记

包含Vue学习笔记（路由、Vuex、Vue知识点整体回顾总结）

### （一） Vue学习笔记——路由

\*\*\*\*\*\*vue简单介绍\*\*\*\*\*\*

\*\*Vue-路由篇router\*\*

vue用来实现SPA的插件

安装：npm install -g vue-router

一、使用vue-router

1 创建路由器: router/index.js

import VueRouter from 'vue-router'

//使用路由器

Vue.use(VueRouter)

//导入路由对应得组件

import About from '../views/About.vue'

export default new VueRouter({

routes: [

{ // 一般路由

path: '/about',

component: About

},

{ // 自动跳转路由

path: '/',

redirect: '/about'

}

]

})

2. 注册路由器: main.js

//引入index.js

import router from './router'

new Vue({//配置对象的属性名都是一些确定的名称，不能随便修改

router // 注册路由器

})

3. 使用路由组件标签:

<router-link to="/xxx">Go to XXX</router-link>

<router-view></router-view>

二、编写路由的3步

1. 定义路由组件

2. 映射路由

3. 编写路由2个标签

三、嵌套路由

children: [

{

path: '/home/news',

component: news

},

{

path: 'news',//简化写法

component: news

}

]

三、向路由组件传递数据

//路由链接传递

params: <router-link to="/home/news/abc/123">

//路由组件标签传递

props: <router-view msg='abc'>

params示例说明：Message.vue向MessageDetail.vue传id

a)路由配置——路由组件编写

使用占位符:id

children:[

{

path:'/home/message/messageDetail/:id',

component:MessageDetail

}

]

b)在Mesagae.vue中使用路由链接传递

<router-link :to="`/home/message/messageDetail/${message.id}`">{{message.title}}</router-link>

c)在向MessageDetail.vue中获取

{{$route.params.id}}

四、缓存路由组件

<keep-alive>

<router-view></router-view>

</keep-alive>

五、路由的编程式导航

this.$router.push(path): 相当于点击路由链接(可以返回到当前路由界面)

this.$router.replace(path): 用新路由替换当前路由(不可以返回到当前路由界面)

this.$router.back(): 请求(返回)上一个记录路由

### （二）Vue学习笔记——Vuex

\*\*\*\*\*\*vue简单介绍\*\*\*\*\*\*

\*\*Vue -Vuex篇\*\*

一、 vuex是什么

github站点: https://github.com/vuejs/vuex

在线文档: https://vuex.vuejs.org/zh-cn/

简单来说: 对应用中组件的状态进行集中式的管理(读/写)

二、状态自管理应用

state: 驱动应用的数据源（data）

view: 以声明方式将state映射到视图

actions: 响应在view上的用户输入导致的状态变化(包含n个更新状态的方法)(函数)

三、多组件共享状态的问题

多个视图依赖于同一状态

来自不同视图的行为需要变更同一状态

以前的解决办法

\* 将数据以及操作数据的行为都定义在父组件

\* 将数据以及操作数据的行为传递给需要的各个子组件(有可能需要多级传递)

vuex就是用来解决这个问题的

四、4. vuex的核心概念

1). state

vuex管理的状态对象

它应该是唯一的

const state = {

xxx: initValue

}

2). mutations

包含多个直接更新state的方法(回调函数)的对象

谁来触发: action中的commit('mutation名称')

只能包含同步的代码, 不能写异步代码

const mutations = {

yyy (state, {data1}) {

// 更新state的某个属性

}

}

注意：把数据data1放在{}中获取

3). actions

包含多个事件回调函数的对象

通过执行: commit()来触发mutation的调用, 间接更新state

谁来触发: 组件中: $store.dispatch('action名称') // 'zzz'

可以包含异步代码(定时器, ajax)

const actions = {

zzz ({commit, state}, data1) {

commit('yyy', {data1})

}

}

注意：把数据data1放在{}中传递

4). getters

包含多个计算属性(get)的对象

谁来读取: 组件中: $store.getters.xxx

const getters = {

mmm (state) {

return ...

}

}

5). modules

包含多个module

一个module是一个store的配置对象

与一个组件(包含有共享数据)对应

6). 向外暴露store对象

export default new Vuex.Store({

state,

mutations,

actions,

getters

})

7). 组件中:简化写法

import {mapGetters, mapActions} from 'vuex'

export default {

computed: mapGetters(['mmm'])

methods: mapActions(['zzz'])

}

{{mmm}} @click="zzz(data)"

8). 映射store

import store from './store'

new Vue({

store

})

9). store对象

1.所有用vuex管理的组件中都多了一个属性$store, 它就是一个store对象

2.属性:

state: 注册的state对象

getters: 注册的getters对象

3.方法:

dispatch(actionName, data): 分发调用action

### （三）Vue学习笔记——Vue知识点整体回顾总结

\*\*\*\*\*\*vue知识点整体回顾总结\*\*\*\*\*\*

\*\*Vue总结篇\*\*

一、基础语法

基础认知(是什么，安装，环境搭建)

各文件层次结构(main.js--->App.vue--->子组件)

1.一般指令

{{}}

v-bind:prop/:prop

v-model

v-text

v-html

v-if

v-else

v-show

v-for

2.事件指令

v-on:prop/@prop

@click

@keyp.center

事件修饰符：.prevent,.stop

3.class与style绑定

4.表单输入绑定

text/textarea v-model="content"

checkbox v-model="likes"

radio v-model="sex"

select v-model="cityID"

5.vm

data

computed

methods

mounted 异步/定时器

二、组件化编码

使开发更加有条理，提高团队效率

流程：拆分页面--->静态组件--->动态组件--->css拆分--->初始化数据显示--->动态交互

组件的使用：引入组件--->映射成标签components--->使用标签组件

组件通信(常用)：props pubsub vuex

三、ajax

记得在需要使用的组件中引入对应的资源

vue-resource

this.$http.get(utl).then((response)=>{

success

response.data

},(response)=>{

fail

response.statusText

}

)

axios

axios.get(url).then(response=>{

success

response.data

}).catch(error=>{

fail

error.message

})

四、UI组件

ele

五、路由router

管理请求

目的：分发请求，管理框架组件

使用流程：创建路由--->注册路由main.js--->使用路由<router-link to=""> <router-view>

六、Vuex数据共享管理

其实就是将数据放在一个公共的地方，需要的组件可以直接操作数据

1.store组成结构：

state

mutation

actions

getter

new Vuex.Store({

state,

mutations,

actions,

getters,

})

2.使用流程：

创建Vuex.Store(store/index.js)--->注册store(main.js)--->组件.vue调用--->actions--->

mutations

3.补充：

mutations{//只能写同步代码，不能写异步

mutation(state,{参数}){

}

}

actions{//可写异步代码，ajax,定时器

action({commit,state},参数){

commit('mutation',{参数})

}

}

getters(){

getter(state){

}

}

4.组件中使用方法：

引入：import {mapGetters,mapActions} from 'vuex'

this.$store.dispatcher('action')--->简写 ...mapActions(['action','action2',...])

this.store.getters('getter')--->简写 ...mapGetters(['getter',...])

注意：简写可以直接省略方法名/书写

如：

methods:mapActions(['actions'])

computed:mapGetters(['getter'])

七、源码分析

基础准备

数据代理

组件编译

数据劫持/数据更新