初始化项目：软件和安装环境

nodejs配置 vue和脚手架配置 MongoDB数据库下载 VScode编程软件

使用express后端框架，elementUI进行界面设计

安装nodemon依赖包

首先使用github创建新的仓库，本机还需要配置git和公钥，配置.gitignore

First\_Vue\_NodeJS\_Project

非关系型数据库和关系型数据库的区别

使用elementUI、nodejs、vue、MongoDB实现的全栈测试项目。

全部环境配置完毕后进行项目目录搭建

vue create 创建web 和 admin 配置安装依赖包

三个端，server负责提供后台接口和前台接口

package.json 文件script配置自定义启动指令 "serve":"nodemon index.js",

P4：admin后台界面，element引入 vue add element，

安装完毕后，加入路由vue add ，不选择history模式

创建主要的父组件Main在admin/src/views下，直接引用官网上的后台界面模板放入组件中

router/index.js路由中引入

100vh：撑满整个浏览器高度

在左侧菜单列表中加入element封装的router属性，进行跳转，设置选项的跳转路由

在router中的mian路径中添加子路由

新建需要跳转的页面CategoryEdit.vue，点击左边菜单，main中右方主界面会更换成为该组件

mian中的主界面页面设置为router-view，根据路由显示组件

设计CategoryEdit中的一些组件内容

安装谷歌浏览器的VUETOOLS调试工具，可以即时查看vue的data中的数据变化（未安装）

@submit.native.prevent="save"提交表单，使用原生的submit，且阻止跳转页面，执行save()

请求一个接口提交数据

安装配置axios : npm i axios

编写自定义的axios，加入指定的前端接口地址baseURL: 'http://localhost:3000/admin/api'，编写http接口并引入到main.js（vue对象文件）中，加载到Vue的原型对象中

可以在任意情况使用this来使用这个封装好的axios实例

p6:服务端创建分类

npm i express@next 加入依赖包 最新的5.0版本（使用http-assert必须要5.0支持async异常处理）

再加入mongoose 数据库依赖 cors跨域依赖（多看看跨域的定义和其他跨域方法）

index.js创建服务器和监听端口

服务器端分类路由建立对应admin和web的路由文件夹

require('./routes/admin')(app)

编写admin端的路由，其中涉及了async的异步知识（多看看）

创建数据库连接和数据库对象导入，新建plugins文件夹存放db.js文件

require('./plugins/db')(app)

学习路由和模块的插件写法

module.exprots = app =>{

//引入MongoDB数据库对象或其他模块，分类编写

//最后引入到服务器index.js中

}

此处为MongoDB数据库知识，多了解一下比如mongoose

新建模型文件夹models

文件夹下进行分类创建Category.js

编写完毕后导出模型

并在admin端的index路由进行引用

模型用于接收数据库传回的json数据并转换为模型定义的字符串类型

需要导入部分中间件（顺便导入跨域中间件）

中间涉及了异步处理async和await（多看看视频）

此时后台的admin分类接口定义好了

就是app.use('/admin/api',router)及其下定义的各个接口

此时测试前端CategoryEdit.vue发起向后端接口的请求，save()方法

使用封装好的axios实例调用post传数据给数据库

再返回接收返回数据（异步处理），再跳转到分类页面

//(2). 使用程序跳转:

// this.$router.push("/相对路径")

// $router: router路由器对象进入new Vue()后，就被改名为$router。$router就指路由器对象。$router能执行跳转动作！

此时测试已完毕

p7：实现新建分类后。分类列表页显示数据的实现

实现点击分类列表或者新建分类后跳转到该页面同时加载数据库数据

admin下的views新建CategoryList.vue

路由中引入该vue

弹幕说可以使用懒加载（多看看东哥视频）

新建组件样式（复制ui）

定义item保存数据库返回的数据

设置element组件的一些属性，使item数据显示

定义加载数据的方法methods：fetch()使用异步调用后台接口，返回数据库数据到item中

此时回到server后端编写分类页的接口

使用get，使用Category数据库实例模型调用数据库函数find()显示所有数据，同时设置数量限制

返回到admin端

p8:分类页的信息编辑

在分类列表添加编辑按钮同时设置路由跳转并传参，使用模板字符串和插槽组件获取本行id跳转，地址前加上斜杠从根目录开始

@click="$router.push(`/categories/edit/${scope.row.\_id}`)"

=>

<http://localhost:8080/#/categories/edit/5e19a0da319bb005586955aa>

admin的路由中引入edit/:id，根据不同传入的url路径使用同一个CategoryEdit页面组件

添加props使得CategoryEdit组件可以使用传入的id参数，建议组件中的props解耦改成数组模式对应官方文档

不需要再截取URL进行获取id

通过三目运算符判断有无id更改组件中h1标题的显

<h1>{{id?'编辑':'新建'}}分类</h1>

同时判断是否有id，在跳转的时候created()执行新的组件加载函数（短路判断）

this.id && this.fetch();

编写加载编辑页面的函数

async异步调用接口，返回数据到model中，model再返回到input中。

对编辑页面的数据进行修改的axios使用put而不是post或者get

ESlint 代码规范（看看怎么修改，大牛才开）

根据有误id执行不同的方法（区分新增和修改）

if(this.id){修改方法a}else{新增方法b} 实现并存和组件共用

await this.$http.put(`categories/${this.id}`,this.model)

再去服务端admin下路由新增修改的接口put（修改查询语句，根据d）

put需要根据查找条件同时传递请求主体req.body并修改（传入两个参数）

弹幕发现BUG:从list页面点编辑按钮进入编辑页面后，点击菜单的创建，原来传递的id还在，会因为逻辑问题（已存在）报500错误。（id因为路由跳转没了，但是model还在）

（弹幕推荐可以使用路由守卫解决）

因为是同一个组件，id是已经保存好的了。此时跳转回去

url的完整结构：

<scheme>://<user>:<upwd>@<host>:<port>/<path>/<params>?<query>#<flag>

<user>:<upwd>：用户名和密码

<host>：指主机名称，资源所在服务器的IP地址，或者域名（需要DNS转换为IP地址）

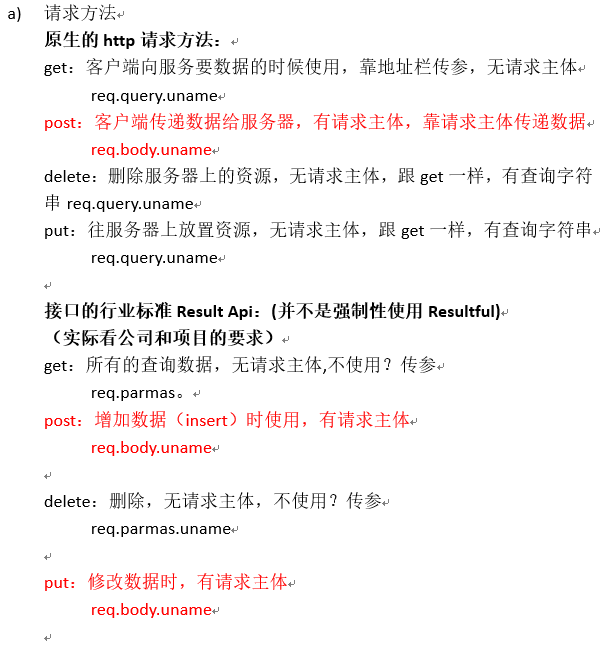
<port>：端口号，每项服务器在服务器上都对应一个监听端口号

<path>：资源在服务器上具体的路径结构

<params>面试会提及，项目也会使用，参数，跟踪状态参数 session/cookie

<query>：查询字符串，get方法带参提交时的查询字符串

<flag>：锚点



了解mongodb的查询语句

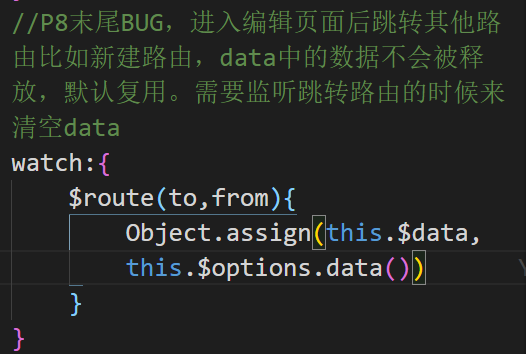
post和put的区别，请求主体的意义多看看

以及req的请求方法parmas和query和body的区别。 以及路由传参！！！

<https://www.jb51.net/article/163764.htm>

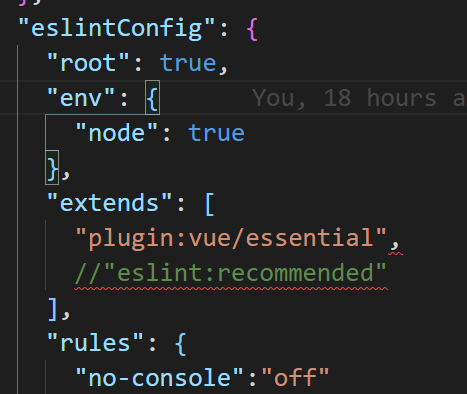
路由守卫：（暂时还没用上，只是清除了同一组件路由跳转时data清空）

当组件重用，内部路由切换时，created不触发，data数据也会保留





<https://blog.csdn.net/creabine/article/details/81664203>



注释掉该行代码可以去掉eslint的报错（坑爹手贱）

p9:分类删除

ui组件中添加删除按钮，添加remove函数执行删除（添加一个确认框以避免误操作）

在确认的UI组件中添加异步函数调用删除接口，传递row中的\_id参数，最后重新加载列表

后端接口await Category.findByIdAndDelete(req.params.id,req.body)

p10:子分类

不同分类设置自己的父子字段，形成一个链式结构区分

上级分类名不能写死。否则子分类的父子字段全部需要更改

所以我们就使用id进行关联

在新建分类中添加上级分类的下拉菜单进行选择，新增加载组件时自动将上级分类数据加载到菜单中（项目暂时使用之前get获取列表的接口，可以自己另外新建接口加载父分类）

created()加入父分类加载的函数

UI组件中设置父分类选项，使用v-for遍历数据形成选项

<!-- label是选项显示的内容，value是点击,v-for循环遍历选项 -->

选择好上级分类后提交创建子分类，此时需要去后端更改一下接口

因为新增了一个分类关联的字段，需要在后端的数据库模块中新增parent，同时设置关联属性。

点击新增分类，自动创建一个新的子分类（包含了关联字段，是某个分类的子分类）

子分类中的parent属性关联父属性的id，可以动态将父分类的属性比如name传入并使用（很像mysql的关联查询），同时指定关联的模型，此时模型中全是分类，所以是自身模型。

parent:{ type:mongoose.SchemaTypes.ObjectId,ref:'Category' },

（增加的分类都是存在一个表中的。MongoDB多了解下）

修改显示的id转化为name

找到显示列表的接口进行编写

在获取子分类的parent属性时添加一个函数populate()使得返回的数据中，parent数据不是单纯的id而是整个的父分类的对象！这样前端就能直接调用对象中父分类的parent.name属性而无需大费周章。

p11：crud通用接口 :增删改查通用接口，只更改路由以及数据库模型，增删改查都是通用的（后端）

任何类型的资源都需要有一个数据库模型比如英雄、物品等模型

前端要修改某个类型时，后台接口的接口地址要动态改变，增删改查详细就不变

首先在底部大接口进行修改，增加一个动态参数/rest/:resource，前缀rest表示这是通用的增删改查的接口，resource来匹配各个接口中的接口名（resource就是前端调用接口返回的类型名）

然后删除上面接口的接口名，空的添加/（得到一套新的通用接口）

同时还要配置通用数据库模型代码和其导入。

重新编写模型导入

删除固定的头部导入

在每个接口函数中导入通用的模型

const Model = require(`../../models/${req.params.resource}`);

需要在router常量中定义mergeParams:true

使得底部父级路由的url可以合并到子接口中的url中进行调用，否则内部是无法访问app.use中定义的参数resource的。也就无法根据地址导入模型

然后通过一个包inflection调用函数Classify更改，将前端获取到的模型名改为大写开头并改为单数，使模型包地址正确而导入成功（其实改一下模型名也可以（艹））

前端也可以进行封装路由和axios，见章哥的串讲1.14

然后将这段模型导入做成异步函数async，放入父接口中，挂载Model（导入包）到req对象中，这样子接口就可以直接使用req.Model执行数据库操作。

（其实就是自定义中间件emmmm封装）

next()异步函数执行完后使下一个函数执行。去理解一下async

populate函数可能其他Model并没有用上，需要添加逻辑判断是否使用。

使用了setOptions方法



p12:物品装备管理

创建物品装备的组件vue文件，同时修改main中的菜单列表跳转路由内容

删除不需要的函数方法，修改接口地址为items

在admin的路由字典中导入路由，设置子路由属性

新建ItemList.vue待用

进入新建物品页面，此时后端因为设置了通用路由，接口都是通用的

此时后台的item的模型还没建立

新建Item模型添加icon属性（图片地址）

ItemList.vue中将分类的接口名更换为物品的接口名items

完成物品的增删改查

p13：图片上传

在图片新建页面中添加图片上传el-upload的UI组件，修改组件的属性

此处使用的是头像上传的组件，使用的类名都是头像相关可以自己修改

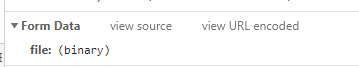
icon就是图片在后台的地址

上传图片的原理就是发起异步请求将图片传给某个接口（接口保存这个图片然后返回一个地址给前端显示），需要地址

:before-upload="beforeAvatarUpload"该属性时在上传前执行的函数用于判断图片格式大小等

先在写好上传到后端的地址:action="$http.defaults.baseURL + '/upload'"

因为是动态的需要加:在action前，因为这个组件时使用自带的ajax请求，所以要写全地址

已经将图片转换为二进制了

一个file

后端写上传接口

因为之前的路由都是进行增删改查的，此时我们在后端只能新建一个上传的路由了

然后安装一个负责处理上传数据的中间件，npm i multer

导入multer包，设置图片的保存地址常量，放入接口中

用这个地址去接收前端返回过来的file

这个file名字是个可自定义名字的参数，可以在elementUI 的el-upload的上传组件中修改

托管静态文件路由，使得前端可以通用浏览

在接口中将后台图片的url拼接好然后插入到传过来的file中，再返回给前端

file.url = `http://localhost:3000/uploads/${file.filename}`

此时上传图片，后端已经返回给前端数据了，但是因为Vue的响应式问题，执行函数强行新增data中数据的属性时，有可能加载不到data中

此时再使用Vue内部的方法，直接给数据添加属性

此时已经可以显示了，同时也能上传icon属性到数据库模块中进行新增或者修改

再添加列表中显示图标的功能，添加插槽UI组件，传入icon的地址并显示

p14:英雄管理

main.vue左侧添加英雄的选项

新建edit和list页面修改

admin路由新增英雄

新增英雄此时接口是有通用接口的，但是后端数据库模型还未定义

新建Hero模型，增加头像地址属性avatar

此时前端可以新建英雄了，传入新英雄的数据

修改list，头像地址的属性avatar放入组件UI中

p15:英雄的编辑界面（模型字段）

在模型字段中如果一个属性中还有属性，最后使用数组包裹

因为可能会使用多个整个属性，或者多次使用

但是如果这个外层属性只使用一次，则可以使用对象包裹（多写一遍结合视频去理解）

英雄里面使用到了装备的模块，需要像导入分类id一样导入装备或者物品的id或者整个对象

铭文和装备使用是一样的，到时候看情况添加

p16:英雄编辑（admin）

新增和编辑界面添加上一节添加的属性所对应的组件UI

select加载的是对象而不是数组，如果加载的是数组（多选）则需要加关键字multiple

其中需要使用el-option组件，使用v-for遍历传过来的categories分类数组中的内容

<el-option v-for="item of categories" :key="item.\_id" :label="item.name" :value="item.\_id"></el-option>

data中预先定义categories空数组存放传回来的分类信息

在created中调用加载分类的数据将分类categories传入

以后设计到模型属性中还有属性，多层级的情况，获取返回到前端的数据中时，要先在data中定义好。

因为后端返回到前端的res.data会直接覆盖掉之前定义好的data，所以需要使用Object.assign()方法并存。（此处先定义好）

然后选择多选装备来选择，采用的还是之前选分类那样，只不过选项中遍历的对象是装备的模块

method中添加加载装备的异步请求

在进入编辑页面时created调用一次加载函数

然后添加后面的使用技巧，添加一个文本UI组件

P17：技能编辑（UI）

使用elementUI的el-tabs组件创建选项卡操作value表示当前显示的选项卡

el-tab-pane子选项卡标签，name属性是value的取值

多多看elementUI的文档多熟悉去使用！！！！

for of遍历数组 for in遍历对象（v-for的使用方法回去看笔记！！）

<要反复生成的元素 v-for="(elem, i) of 数组" :key=”i”> i此时是对象或者数组的索引值

i从0开始索引

此时是新增技能，点击添加技能生成一个模板，所以我们需要使用循环遍历skills数组，并且定义点击事件往skills数组中push空数组以生成新的输入框。（建议提在data前定义好skills）

在循环添加的技能模板中，增加图标上传（将头像上传的模块复制过来），还有描述文本框

修改上传图片的属性中的data值为传过来的技能模型中的属性，名称和描述，小提示同理

P18：技能编辑（交互）

更改上传技能图标的函数（跟上传头像的url不一样）

这里up直接不在方法中写了，直接在标签属性中写箭头函数

将后端传回来的url地址直接赋予了data中skills的icon属性（为了避免Vue不及时加载或者未找到传过来的model中的的属性，可以用$set方法或者在data的model中预先定义好）

将其他描述和名字之类的传入属性修改好

新增技能

如果多加了几个空的技能模板传入了数据库

可以使用删除，新增删除按钮添加点击删除的事件

@click="model.skills.splice(i,1)"

删除当前技能

P19：文章管理

main.vue中添加文章的左侧子菜单，修改本地跳转的路由，新建文章修改和列表vue

因为文章可能有多个分类，所以使用多选的分类选择框，修改对应属性和函数调用接口

修改created加载时将分类数据传入然后遍历到选项菜单中

新增后端文章的数据库模块

修改list中的一些属性，完成文章的新增和删除修改

P20:富文本编辑器

需要先npm install --save vue2-editor安装依赖包

在admin端上加

根据文档添加导入包和子组件添加

添加组件标签到详情下

使用这种编辑器可以直接编写很多样式的文本包括插入t图片等，但是我们需要使用自行编写的图片上传功能，保存的是图片的后端地址，下一节说明