目录

[Mongodb基础知识 1](#_Toc9083497)

[启动流程 1](#_Toc9083498)

[连接相关 1](#_Toc9083499)

[Mongodb权限配置 1](#_Toc9083500)

[最初安装的mongodb 1](#_Toc9083501)

[配置超级管理员 1](#_Toc9083502)

[Mongodb常用指令 3](#_Toc9083503)

[Mongose使用 3](#_Toc9083504)

[基本属性的构建 3](#_Toc9083505)

[数据库异常处理 4](#_Toc9083506)

[数据库基本配置 4](#_Toc9083507)

[Mongose异步问题 4](#_Toc9083508)

[Mongodb数据库备份问题 5](#_Toc9083509)

[Mongose.model常用操作方法 5](#_Toc9083510)

[Restful风格API 5](#_Toc9083511)

[基于nodejs进行设计 5](#_Toc9083512)

[Express路由配置 5](#_Toc9083513)

[Mongose复杂查询 5](#_Toc9083514)

[只取得想要的fieldName 5](#_Toc9083515)

[POSTMAN使用 6](#_Toc9083516)

[VIM常用指令 6](#_Toc9083517)

[优秀的参考资料 6](#_Toc9083518)

[Require and import 6](#_Toc9083519)

[导出的不同： 6](#_Toc9083520)

[常用库 6](#_Toc9083521)

[Fs文件读取 6](#_Toc9083522)

[nconf 6](#_Toc9083523)

[VSCode常用快捷键 6](#_Toc9083524)

[编辑 6](#_Toc9083525)

[CSS布局经验 7](#_Toc9083526)

[B站用户信息弹出层 7](#_Toc9083527)

[Icon与文本的对齐问题 7](#_Toc9083528)

[ 7](#_Toc9083529)

[需求 7](#_Toc9083530)

[需求的原因： 7](#_Toc9083531)

[原型 8](#_Toc9083532)

[原型的原因 8](#_Toc9083533)

[各种居中情景的解析 8](#_Toc9083534)

[父元素为块，子元素为行内元素，如<b> <span>,内部是文本 8](#_Toc9083535)

[使用icon 8](#_Toc9083536)

[外部UI框架与自身定义样式冲突 8](#_Toc9083537)

[Canvas 8](#_Toc9083538)

[经验 8](#_Toc9083539)

[加密解密 9](#_Toc9083540)

[Md5加密 9](#_Toc9083541)

[AES加密 9](#_Toc9083542)

[RSA加密 9](#_Toc9083543)

[Crypto使用 10](#_Toc9083544)

[express router 10](#_Toc9083545)

[中间件 10](#_Toc9083546)

[app.use 与 router.use 10](#_Toc9083547)

[VUE高级 10](#_Toc9083548)

[.sync 10](#_Toc9083549)

[异步 10](#_Toc9083550)

[Promise 10](#_Toc9083551)

[Axios 10](#_Toc9083552)

Mongodb基础知识

启动流程

运行net start mongodb 启动win服务

运行mongod

可携带指令查看：mongod –help

职责是用来启动服务器，并进行配置。配置包括监听的端口号，数据目录等

一般在安装的bin目录下配置mongo.config文件，作为mongod的参数

运行mongo，进入命令行模式，此模式下就可以操作数据库了

连接相关

Connection string URI格式

格式1：mongodb://localhost:27017/admin

格式2：mongodb://username:password@localhost:27017/admin

Mongodb权限配置

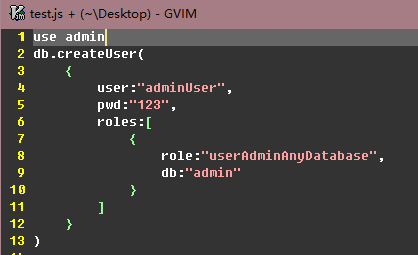
最初安装的mongodb

安装后的mongodb是没有权限验证的，任何人都可以通过默认的27017端口访问。

基本的访问格式是：mongodb://localhost:27017/databasename

配置超级管理员

进入admin表中创建管理员用户：



退出，运行 mongod --auth开启访问控制，或者，在启动配置文件mongo.config中配置



通过管理员进入数据库：

方法1：先进入数据库再验证



方法2：进入数据库时验证



必须先进入过admin表，然后再use进入其他表，才有超管权限！

这个role的权限是创建用户，新建的表，需要为其创建user。

各种user的权限：<http://www.cnblogs.com/SamOk/p/5162767.html>



Mongodb常用指令

Show dbs 显示所有数据库

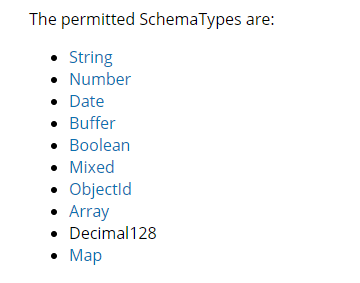
Use database 切换到一个数据库，若没有，则会创建

Mongose使用

基本属性的构建

New schema:需要一个js对象作为参数，内部key是字段名，value是此字段的属性。

构建schema时，字段可配置的schema types：



其他可配置的各种options：

<https://mongoosejs.com/docs/guide.html>

而mongose.model函数，调用时的参数：(modelname,schema实例),其返回值，一般导出，猜测其返回了一个构造函数，在后续使用时，需要new modelname(),参数是要设定的符合schema规范的一个js对象，也就是要存入的数据。此modelname构造出的实例，可以调用save来进行数据库操作了。

这里要注意，model构建时，第一个参数的名字，是要映射的collection的名字，该和数据库的collection的名字对应。

这个参数一的名字，会被强行加上s，和数据库collection名不匹配，因此，一般加参数三，名字必须完全一致，才能找到对应collections

Save需要new modelname,而find不需要new，直接用model就行。

数据库异常处理

调用save函数，需要传入回调函数作为参数，回调函数接收参数err，可以通过这来决定向前端返回的信息。

数据库基本配置

连接

Connection stirng

连接成功和失败时的回调

路径存在哪个文件？使用nconf，存入一个json文件中

问题，业务使用时，引入schema还是model好？引入model

Mongose异步问题

Mongose的model操作是异步的，想将操作封装后导出，会遇到异步问题。

解决1回调：model.save({},func),这里的func可以作为封装的参数，由调用者填入。

解决2promise：model.save({}).exec();返回promise给调用者。



Mongodb数据库备份问题

Mongose.model常用操作方法

<https://mongoosejs.com/docs/api.html#Model>

Restful风格API

基于nodejs进行设计

必然，每一个数据库中的document，都对应着建立一个schema，因此，有几个表，就要建立几个schema并实例化为model，

Express路由配置

Mongose复杂查询

只取得想要的fieldName



POSTMAN使用

一个API调试、Http请求的工具

用来设计前后端接口规范

新建request

在request中建立example表示response

VIM常用指令

选中多行: ctrl+v,进入视图模式。

移动到行首： ^

移动到行末： $

优秀的参考资料

Mongose schema配置：<https://www.jianshu.com/p/d073a385d282>

Require and import

导出的不同：

Module.exports=target; 用于require

常用库

Fs文件读取

nconf

VSCode常用快捷键

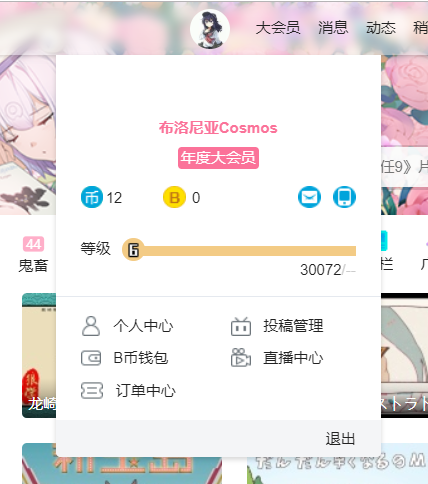
编辑

Home：到行首

End: 到行尾

CSS布局经验

B站用户信息弹出层



看到布局，先着眼于大的块划分

Icon与文本的对齐问题



需求

需求的原因：

找到一种方式，逼迫自己，让自己对于所需要的功能，和性能，做到最极致，透彻的理解，成为产品的理解者。

不论写文档，画图，成为理解者这一目的，是不变的。

方法

总结需求，然后一个一个实现

原型

原型的原因

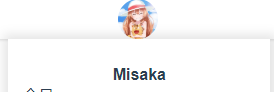
为布局所苦恼，为动画所迷茫，不断纠结不断修正，难以获取想要的样子。

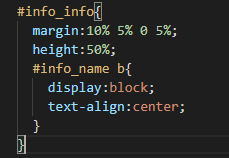
所以，画出原型，作为蓝图，让混乱繁杂的css，变得明晰。

各种居中情景的解析

父元素为块，子元素为行内元素，如<b> <span>,内部是文本

设置子元素display:block; text-align:center;子元素宽度扩展到和父元素一致，同时使用文本对齐，可以居中文本。





使用icon

外部UI框架与自身定义样式冲突

Canvas

经验

1 尽可能让组件小型化，否则单文件中出现大量html，js ,css难以查找，难以管理，例如复杂导航栏，可以将头像部分拆分。

2 异步操作的方法调用，结果处理，与vuex的store怎样结合？以登陆为例

先考虑，怎样的数据，才需要放在store中呢？全局通用，比如登陆状态，比如用户信息。

而画廊的图片数组，仅仅在画廊这个页面级组件中使用，因此，其通过api获取数据这一过程，在其自身created内调用即可，而不需要在store的action中调用。

3 对于一次迭代，不要写一点前端，再写一点后端。需要把这次迭代的接口设计完成，专心一个领域，最后再联调。

4 跨域proxyTable的写法<https://blog.csdn.net/aydongzhiping/article/details/81102940>

5页面刷新<https://blog.csdn.net/qq_16772725/article/details/80467492>

<https://blog.csdn.net/yangxiaodong88/article/details/81387672>

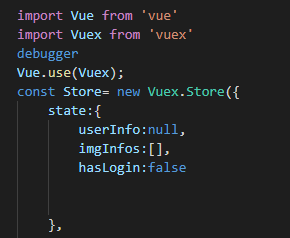
6 store的设计，其中配置需要返回store实例才行，而不是返回一个配置

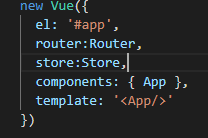
7 import是单例的，new发生了什么？

8 mainjs实例创建，app.vue根组件，router模块，store模块加载顺序。

9 函数若不执行，则就不存在this上下文，上下文是运行时创建的，运行后销毁的。

10 安装插件时，配置插件时，记住要用键值对，记住，键的名字是固定的，不能自定义。例如配置





上图中，可以只写一个store，但是这必须是值变量，否则，值另起名字，需要写出完整的键值对，而键名字必定是store，值名字自己定。

参考：<https://blog.csdn.net/marswill/article/details/74368939>

11 运行时的区域：函数区域，函数运行区域，对象区域，new本质也是object

12 项目设计中，要进入页面后使用api获取用户信息，然后存储，这次调用放在哪里？

放在路由钩子中，获取数据后，放在store中。而各个页面唯一的依赖store中的userinfo，当其为Null,有两种策略，加载模拟数据，或者，不加载组件

加密解密

Md5加密

**使用场景**：密码存入数据库，防止出现密码明文。令用户密码明文只会出现在用户自己输入时。

**特性**：是不可逆的，加密后无法解密。

**VUE项目中的使用地点**：

**自身封装**：

**加密**

位置：utils.js

**解密**

**无**

**缺陷：**

AES加密

密钥来源：

参数配置解析：

RSA加密

密钥来源

加密及签名流程

数字证书

密钥职责：

公钥：用来加密和验证签名

私钥：用来解密和签名

签名目的：防止数据被篡改

加密目的：防止信息被窃取

实际中，一般是通过RSA加密AES的密钥，传输到接收方，接收方解密得到AES密钥，然后发送方和接收方用AES密钥来通信

默认输出是16进制，但是服务器解析时候是需要Base64的

Crypto使用

<https://blog.zhengxianjun.com/2015/05/javascript-crypto-js/>

中间类型wordarray:

<https://www.cnblogs.com/qiqi715/p/9623421.html>

express router

中间件

<https://www.cnblogs.com/waychan/p/5414749.html>

app.use 与 router.use

二者第一个参数均为路径，实际上是双重匹配，app.use执行第一次匹配，之后，去除匹配的项，再进行二次匹配。这有利于划分路由模块，实现restful风格。

VUE高级

.sync

异步

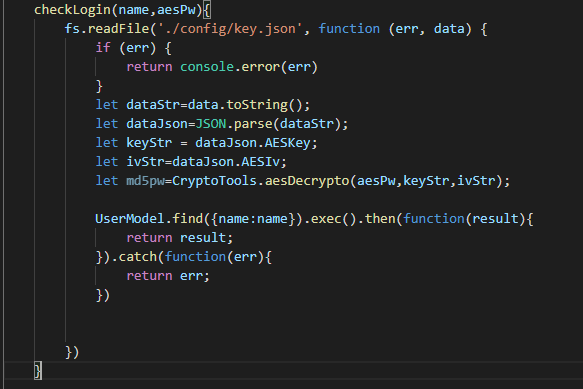
Promise

异步操作的then中，又有异步操作，该如何处理？

对js中异步的思考

先考虑后端，异步出现在db操作和文件操作上，和.net的同步式写法不同，js异步是明确的写出来的，在文件和数据库操作api上，就明确了回调或promise。因此，js的任何异步都要自己来处理，设计，即便数据库操作可以是同步，但依然是异步写法。

很常见的，为了职责明确，你设计一个函数a，它被b调用，a刚好包含了异步操作，这时你的return不能写在异步的回调函数中。



像上图，这个return会直接被跳过，进入b中继续执行，当所有同步执行完成后，才会进入异步队列中取出完成io的操作的回调函数，进入主线程中执行，因此异步必定是在同步流成执行完后，再执行的，因此return被放在了最后。

Then中的回调2 err和catch的区别

Promise不能返回值，只能返回promise

函数都是在其定义的域内执行的，而外界调用仅可以传入参数和获取return。这时回调函数思想的核心。

Then方法的返回值是个新的promise对象。Then方法是同步的，但传入给它的回调函数是异步的，是最后执行的。

最高层的promise传递的参数，只有第一个then能在回调函数中收到，后面的then和最高层promise无关了。

All方法参数是个数组，是多个promise，其均成功时，调用then的异步，那这多个promise如何传给then的回调？

一旦一个promise的状态改变，那就无法逆转了。

Catch是then的语法糖，二者获得的回调函数，不论这回调函数返回值，返回promise，throw error，或者什么都不返回即返回undefined，catch与then都会返回一个promise，关键是这个返回的promise如何构建呢？

Then的职责：

其构成：then方法本身，then获取的两个回调函数，回调函数的参数

当一个promise，p1调用它的then方法时，

Resolve的职责：

设计异步函数API需要的考量：

1 异步api返回的必定是promise

2 使用返回的promise时，会用到then，那么，必须明确这个then监听的promise是谁，它该是最后一个执行完的异步。

3 也必须明确，所监听的promise需要给then传入怎样的数据

4 数据库自带的promise，其resolve不能为我所控制，该如何？可以新建promise对其进行封装，不过使用自己的resolve时要小心作用域，需要转换this

关于异常的捕获及异常信息，多重异步的异常

Axios

返回promise，其then中的response结构：

Data：

发送的request结构

路由

使用js切换路由

<https://www.jianshu.com/p/d4ceee7c769d>

刷新页面的方式及其区别

Middleware思想

Use的使用