**校叮当校园app项目代码清单**



|  |  |  |
| --- | --- | --- |
| 文件状态  【 】草稿  【 】正式发布  【√】正在修改 | 文件标识 | SE-2018春-G15-校叮当校园app需求分析报告 |
| 当前版本 | 0.1.0 |
| 作者 | 陈佳敏、徐毓茜、江亮儒、马益亮 |
| 完成日期 | 2018-5-28 |

**­­­­­**

**版本历史**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本/状态 | 参与者 | 起止日期 | 备注 |
| 0.1.0 | 陈佳敏、徐毓茜、江亮儒、马益亮 | 2018-5-26至2018-5-28 | 完成需求分析初稿，查找模板。 |

**目录**

[一、 概述 4](#_Toc12763)

[1.1需求概述 4](#_Toc26198)

[1.2参考资料 4](#_Toc14103)

[二、 详细设计描述 5](#_Toc26989)

[2.1功能结构（HIPO图、PAD图） 5](#_Toc22285)

[修改个人信息： 9](#_Toc26869)

[三、数据 10](#_Toc10451)

[3.1 E-R图 10](#_Toc13992)

[3.2数据表结构 11](#_Toc23118)

[User表 11](#_Toc10143)

[Commodity表 11](#_Toc30672)

[Order表 12](#_Toc31743)

[Message表 13](#_Toc31139)

[Collection表 13](#_Toc26832)

[四、命名规范 13](#_Toc3818)

[五、编码规则 14](#_Toc32232)

[5.1javascript书写位置 14](#_Toc6268)

[5.2javascript命名规范 14](#_Toc12237)

[5.3注释 15](#_Toc7075)

[5.4严格模式 16](#_Toc19957)

[5.5模式 16](#_Toc29681)

[5.6编写规范 17](#_Toc12249)

[5.7变量 18](#_Toc1150)

[三、 用户模块代码 19](#_Toc30632)

**[[1]](#footnote-1)**

# 概述

## 1.1需求概述

程序将分为以下四大模块：前端信息聚集和检索模块，用户模块，管理员模块，信息库模块。前端信息聚集和检索模块是程序的核心模块之一，它会呈现管理员所要给用户展现的内容以及检索用户需要的信息；用户模块和管理员模块对信息拥有不同的权限；信息库模块更新和存放程序将会用到的数据。

## 1.2参考资料

·《软件工程导论》 清华大学出版社 作者：张海藩等

·SE2017秋-G07-项目计划 作者：陈佳敏、徐毓茜、江亮儒、马益亮

·SE2017秋-G07-可行性计划 作者：陈佳敏、徐毓茜、江亮儒、马益亮

·SE2017秋-G07-需求分析报告 作者：陈佳敏、徐毓茜、江亮儒、马益亮

·SE2017秋-G07-总体设计 作者：陈佳敏、徐毓茜、江亮儒、马益亮

·SE2017秋-G07-详细设计 作者：陈佳敏、徐毓茜、江亮儒、马益亮

·http://blog.csdn.net/baidu\_32262373/article/details/52801121 js编码规范

# 详细设计描述

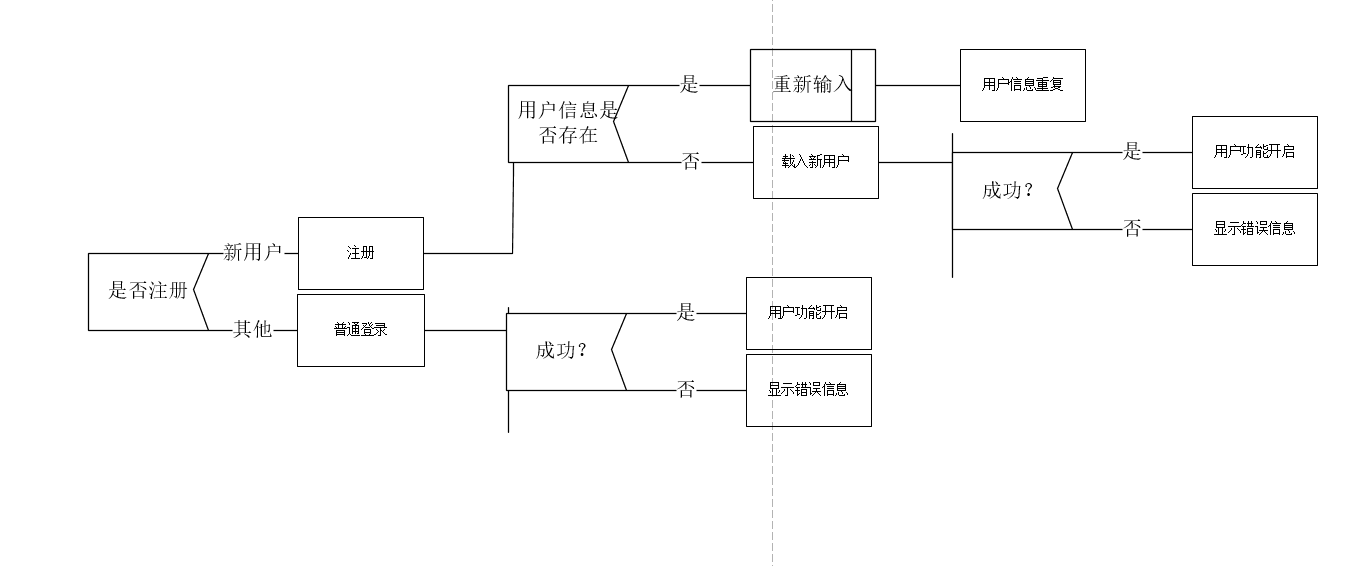
## 2.1功能结构（HIPO图、PAD图）

|  |  |
| --- | --- |
| 系统管理 | 管理员功能：对用户发出警告，从用户信息表中删除非法用户。这项功能只能为管理员所有。 |
| 用户管理 | 这个模块实现了用户请求交易，发布、更新交易信息的功能，并使用户能够更新自己的注册信息以及基本信息。 |
| 交易管理 | 在用户交易管理中，根据用户的提供的交易物品生成了“出售信息表”，用户可以随时对自己发布的信息进行更改或删除。在交易的过程中，系统会根据物品的出售信息目录供用户浏览参考。对于没有明显出售目标的用户，也可以浏览由出售信息表生成的出售物品目录 |
| 出售管理 | 用户发出出售请求后，系统会自动检查用户是否已登录。  用户录入出售信息（包括出售物品名称（gods），出售物品价格（sellprice），出售物品种类（category），出售物品描述（selldes），联系电话（contactp），联系QQ（contactq），联系微信（contactw））。其中物品名称、价格、种类为必填字段。联系方式为电话、微信或QQ，考虑到用户可能不愿意留下自己的电话、微信或QQ，我们将联系方式设为可选字段。当有交易请求时，系统通知用户。  经用户检查录入信息正确无误，生成出售物品信息单。  系统将出售信息纳入出售信息表，并根据出售信息表上的内容发布出售信息 |
| 买卖信息更新管理 | 用户可以随时更新自己的买卖信息。 |
| 搜索 | 用户录入搜索信息开始搜索，系统匹配后生成搜索结果。 |
| 用户信息管理 | 用户信息管理包括更改用户注册手机、更改用户密码、更改用户基本信息；交易管理包括出售管理这些管理操作，主要针对用户信息表进行修改，每个用户只有在登陆后才能修改自己的信息 |
| 更新注册手机 | 用户修改自己的注册手机信息时，必须输入原来的账户信息（注册手机和密码），经系统审核无误后，用户可以修改注册手机 |
| 更新用户密码 | 用户修改密码时，需要输入原密码，并两次输入新密码，经系统审核无误后，用户可以修改密码 |
| 更新用户基本信息 | 用户可以随时更新自己的买卖信息 |
| 用户注册 | 用户录入账户信息，进行有效性检查，生成用户信息表，返回登录界面进行登录 |

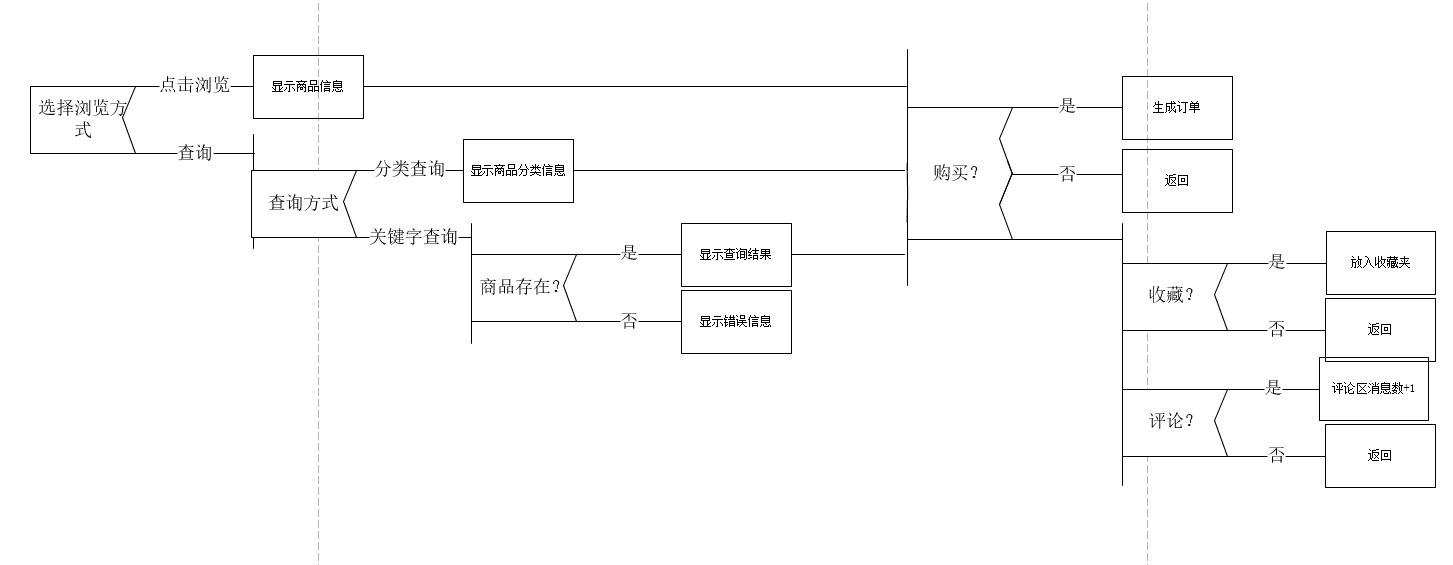
HIPO图

PAD图

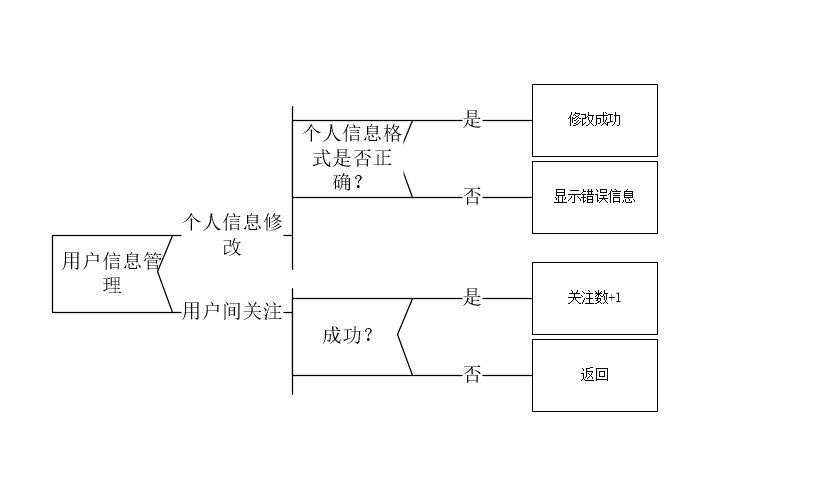
登录注册：



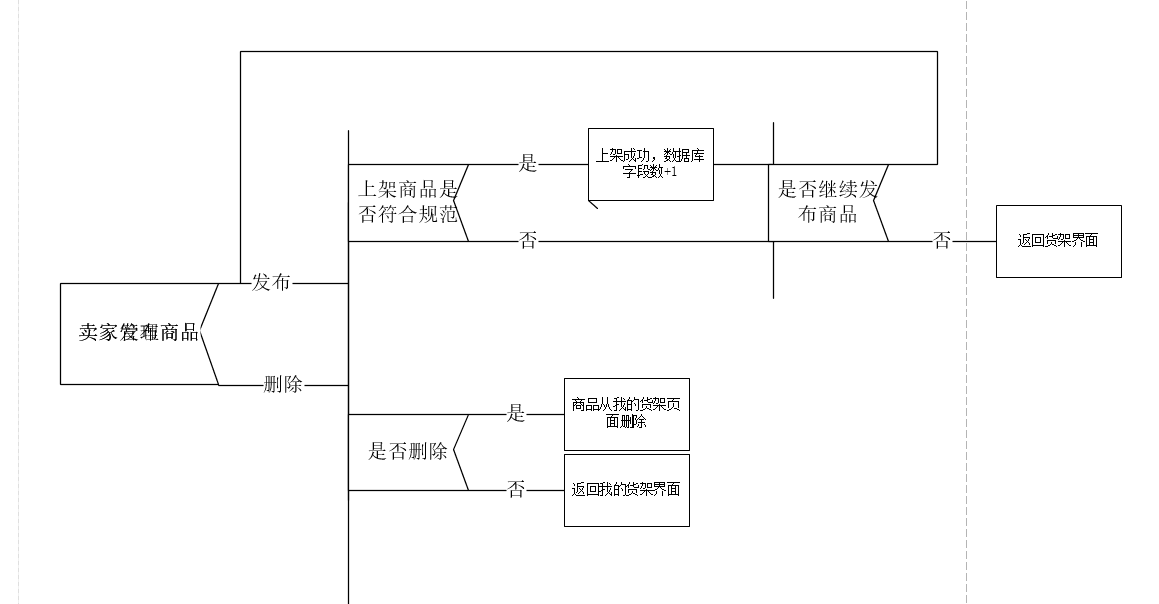
搜索、收藏、购买：



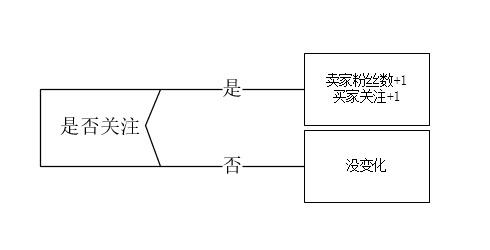
# 修改个人信息：



发布：

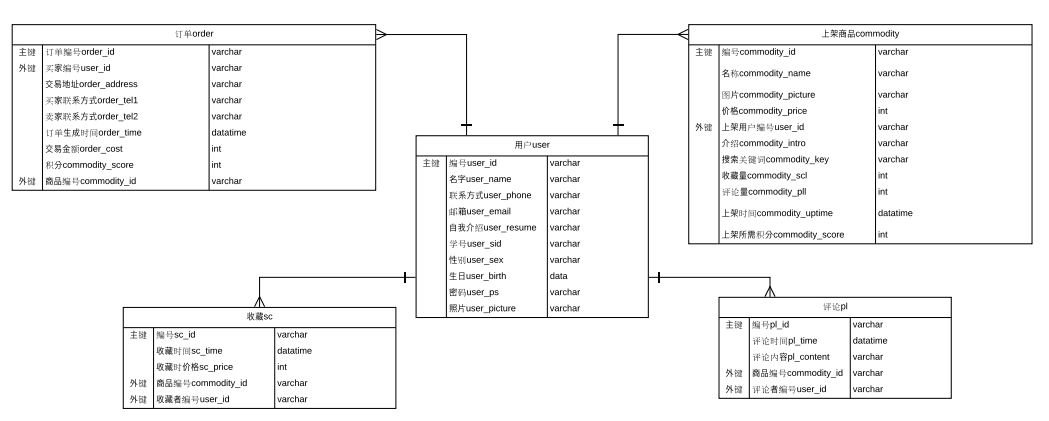


关注：



# 三、数据

## 3.1 E-R图



## 3.2数据表结构

### User表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| User表 | | | | | |
| 字段名 | 实际字段名 | 类型 | 是否能为空 | 键型 | 说明 |
| 帐号 | user\_id | varchar | 不能 | 主键 |  |
| 密码 | user\_pw | varchar | 不能 |  |  |
| 姓名 | user\_name | varchar | 不能 |  |  |
| 性别 | user\_sex | varchar | 不能 |  | 可以设置为保密 |
| 生日 | user\_birth |  | 不能 |  |  |
| 手机号 | user\_phone | varchar | 不能 |  | 用于找回帐号，以及与交易对象进行联系 |
| 邮箱地址 | user\_email | varchar | 不能 |  | 用于找回帐号，以及与交易对象进行联系 |
| 拥有积分 | user\_score | int | 不能 |  | 上架商品需要积分 |
| 学号 | user\_sid | varchar |  |  | 未知 |
| 个人简述 | user\_resume | varchar |  |  | 简单的表述自己 |
| 照片 | user\_picture | varchar |  |  | 数据库只保存图片的地址，图片保存在硬盘里 |

### Commodity表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| commodity | | | | | |
| 商品编号 | commodity\_id | varchar | 不能 | 主键 |  |
| 商品名称 | commodity\_name | varchar | 不能 |  |  |
| 商品图片 | commodity\_picture | varchar | 不能 |  | 数据库只保存图片的地址，图片保存在硬盘里 |
| 商品价格 | commodity\_price | int | 不能 |  | 该价格只做参考，交易双方可私下协商 |
| 上架者帐号 | user\_id | varchar | 不能 | 外键 |  |
| 商品介绍 | commodity\_intro | varchar | 不能 |  |  |
| 商品关键词 | commodity\_key | varchar | 不能 |  | 关键词用于方便商品的查找 |
| 商品收藏量 | commodity\_scl | int | 不能 |  | 商品被用户收藏的数量，该数量作为放在主页的依据之一 |
| 商品评论量 | commodity\_pll | int | 不能 |  | 评论的数量，该数量作为放在主页的依据之一 |
| 商品上架时间 | commodity\_uptime | datetime | 不能 |  | 其他条件相同是，先上架的物品放在主页 |
| 商品上架积分 | commodity\_score | int | 不能 |  | 商品上架积分所需要的积分 |

### Order表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Order | | | | | |
| 订单编号 | order\_id | varchar | 不能 | 主键 |  |
| 买家帐号 | user\_id1 | varchar | 不能 | 外键 |  |
| 卖家帐号 | user\_id2 | varchar | 不能 | 外键 |  |
| 交易地址 | order\_address | varchar | 不能 |  | 双方面对面交易的地址 |
| 买家手机号 | order\_tel1 | varchar | 不能 |  | 也可以是微信等联系方式 |
| 卖家手机号 | order\_tel2 | varchar | 不能 |  | 也可以是微信等联系方式 |
| 下单时间 | order\_time | datetime | 不能 |  | 买家在APP上购买的时间 |
| 交易额 | order\_cost | double | 不能 |  |  |
| 交易花费积分 | commodity\_score | int | 不能 |  |  |
| 商品编号 | commodity\_id | varchar | 不能 | 外键 |  |

### Message表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Message | | | | | |
| 留言编号 | pl\_id | varchar | 不能 | 主键 |  |
| 留言时间 | pl\_time | datetime | 不能 |  |  |
| 留言内容 | pl\_content | varchar | 不能 |  | 内容上限为300个字符 |
| 商品编号 | commodity\_id | varchar | 不能 | 外键 |  |
| 发布者编号 | user\_id | varchar | 不能 | 外键 |  |

### Collection表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Collection | | | | | |
| 收藏编号 | sc\_id | varchar | 不能 | 主键 |  |
| 收藏时间 | sc\_time | datetime | 不能 |  |  |
| 收藏商品的价格 | sc\_price | varchar | 不能 |  |  |
| 商品编号 | commodity\_id | varchar | 不能 | 外键 |  |
| 收藏者编号 | user\_id | varchar | 不能 | 外键 |  |

# 四、命名规范

数据库命名使用匈牙利命名法

匈牙利命名法关键是：标识符的名字以一个或者多个小写字母开头作为前缀；前缀之后的是首字母大写的一个单词或多个单词组合，该单词要指明变量的用途。

编码阶段使用驼峰命名法

驼峰命名法，正如它的名称所表示的那样，指的是混合使用大小写字母来构成标识符的名字。其中第一个单词首字母小写，余下的单词首字母大写。

# 五、编码规则

## 5.1javascript书写位置

### 5.1.1、引入外部 js 文件

格式：<script src=”aaa.js”></script>

注：

html和jsp页面中不允许将后面那个标签去掉

ECMA第五版规定 type 属性不写时默认为 type=”text/javascript”，因此省略这个属性。

### 5.1.2.自己写js 代码

<script>

// 自定义的 js 代码

</script>

### 5.1.3.标签的位置

如果将 <script> 标签放到 <head> 标签中，等待外部js 文件全部下载、 解析并 执行完之后才能解析 <body> 中的元素，这会造成打开页面时会出 现短暂的迟钝现象。

因此我们要求将 <script> 标签放到 <body> 元素中，并且置于 <body> 元素中的html 代码的最下面。

### 5.1.4.推荐

将 html 代码和 js 代码分离，也就是将 js 代码全部写到一个独立的 以 \*.js 的文件中，再通过引入外部js 的方式引入到页面中。可以增加 代码的美观性和方便后期代码维护。

## 5.2javascript命名规范

### 5.2.1.区分大小写

Javascript 中的一切（如：变量、函数名和操作符）都区分大小写。如：变量名 test 和变量名 Test 就是两个变量。typeof 作为关键字不能作为变量名，但typeOf就可以作为变量名了，因为它是一个新的变量。

### 5.2.2.标识符

标识符：变量、函数、属性和函数参数的名字。

推荐命名规则：第一个字符必须是字母或下划线（\_）或美元符号（$）；其它字符可以是字母或下划线或美元符号或数字；用全英文命名的，采用驼峰大小写格式。即第一个字母小写，后面每个有意义的首字母大写。 如：firstSecond myCar标识符不能是关键字、保留字、true、false、null。

### 3.关键字和保留字

在使用中有不熟悉的关键字和保留字可查询下表：关键字有：（javascript 中有意义的单词）ECMA第五版

| 关 | 键 | 字 |  |
| --- | --- | --- | --- |
| break | do | instanceof | typeof |
| case | else | new | var |
| catch | finally | return | void |
| continue | for | switch | while |
| debugger | function | this | with |
| default | if | throw | try |
| delete | in |  |  |

保留字有：（javascript 中暂时没有意义，以后可能被用作关键字的单词）ECMA第三版

| 保 | 留 | 字 |  |
| --- | --- | --- | --- |
| Abstract | enum（枚举） | int | short |
| boolean | export | interface | static |
| byte | extends | long | super |
| char | final | native | synchronized |
| class | float | package | throws |
| const | goto | private | transient |
| double | implements | protected | volatile |
| yield | import | public | let |

## 5.3注释

### 5.3.1.单行注释

// 单行注释

### 5.2.2.多行注释

/\*

\* 这是一个多行注释

\* 多行注释的第二行

\*/

### 5.3.3.方法注释

方法的注释一般应注明方法是干什么用的，注意事项是什么，参数含义， 返回数据信息。

/\*\*

\* 主要功能：生成随机数

\* 注意事项：此方法只能生成11-15位的随机数

\*

\* @param size 生成随机数的长度，范围在数字11~14之间

\* @return number 随机数

\*/

function getRandom (size) {

return Math.random().toString(16).substr(2, size);

}

## 5.4严格模式

'use strict';

严格模式 可以将js 代码中的一些不确定行为得到处理，对某些不安全的操作也会抛出错误，强制性的将js 代码变得更加规范和不容易出错。上述看似字符串的xx 实际上是一个编译指示，用于告诉Javascript 引 擎切换到严格模式。

## 5.5编写规范

### 5.5.1两个空格的缩进

使用2个空格进行缩进，永远也不要使用混合的tab和空格作为缩进。

### 5.6.2换行

使用Unix风格的换行，每行结尾以(\n)结束，永远不要使用Windows的换行符（\r\n）。

### 5.5.3无拖尾空白

永远也不要在一行后面留空格，在提交之前，你要像每顿饭刷牙一样清理你的JS文件。否则，腐烂的气味会驱走贡献者或同事。

### 5.5.4使用分号

根据科学研究，分号的使用是我们社会的核心价值。考虑一下[反对派](http://blog.izs.me/post/2353458699/an-open-letter-to-javascript-leaders-regarding" \t "_blank)的观点 ，但是我们需要传统，不要滥用纠错机制（省略分号）。

### 5.5.5每行最多80个字符

每行最多80个字符。是的，屏幕在最近几年越来越大，但是你的脑子没怎么变，你可以使用多余的空间用来分屏。

### 5.5.6使用单引号

只有在JSON文件中才使用双引号

Right:

var foo = 'bar';

Wrong:

var foo = "bar";

### 5.5.7在同一行写大括号

Right:

if (true) {  
  console.log('winning');  
}

Wrong:

if (true)  
{  
  console.log('losing');  
}

同样，注意在条件前后都加个空格。

### 5.5.8每行声明一个变量

每个var只声明一个变量，它可以更容易地重新排序。但是，并且变量应该在更有意义的地方声明。

Right:

var keys   = ['foo', 'bar'];  
var values = [23, 42];  
var object = {};

while (keys.length) {  
  var key = keys.pop();  
  object[key] = values.pop();  
}

Wrong:

var keys = ['foo', 'bar'],  
    values = [23, 42],  
    object = {},  
    key;

while (keys.length) {  
  key = keys.pop();  
  object[key] = values.pop();  
}

### 5.5.9类名首字母大写

类名的首字母应该是大写的

Right:

function BankAccount() {  
}

Wrong:

function bank\_Account() {  
}

### 5.5.10常量大写

常量应该被声明为普通变量或静态类的属性，全部使用大写字母。

Node.js/V8实际上支持Mozilla的const的扩展，但遗憾的是不能用于类成员，也不是任何ECMA标准的一部分。

Right:

var SECOND = 1 \* 1000;

function File() {  
}

File.FULL\_PERMISSIONS = 0777;

Wrong:

const SECOND = 1 \* 1000;

function File() {  
}

File.fullPermissions = 0777;

### 5.5.11使用 === 操作符

编写不应该只记事规则，还要学会使用。

Right:

var a = 0;  
if (a !== '') {  
  console.log('winning');  
}

Wrong:

var a = 0;  
if (a == '') {  
  console.log('losing');  
}

### 5.5.12早点从函数返回

为了避免if语句的深层嵌套，总是尽可能早地在函数返回值。

Right:

function isPercentage(val) {  
  if (val < 0) {  
    return false;  
  }

  if (val > 100) {  
    return false;  
  }

  return true;  
}

Wrong:

function isPercentage(val) {  
  if (val >= 0) {  
    if (val < 100) {  
      return true;  
    } else {  
      return false;  
    }  
  } else {  
    return false;  
  }  
}

### 5.5.13不要嵌套闭包

使用闭包，但不能嵌套他们。否则，你的代码会变得一团糟。

Right:

setTimeout(function() {  
  client.connect(afterConnect);  
}, 1000);

function afterConnect() {  
  console.log('winning');  
}

Wrong:

setTimeout(function() {  
  client.connect(function() {  
    console.log('losing');  
  });  
}, 1000);

# 六、用户模块代码

## 6.1用户登录

*/\*登录\*/*

function load\_sql(res, account, pw) {

console.log(account + " " + pw);

res.status(200);

res.set('Access-Control-Allow-Origin:\*');

sql = "select user\_id from user where user\_account=? and user\_pw=?";

sqlparams = [account, pw];

connection.query(sql, sqlparams, function (err, result) {

if (err) {

console.log(err.message);

return;

}

var str = false;

for (var a in result) {

str = true;

}

console.log(JSON.stringify(result));

if (str) {

res.end(JSON.stringify(result));

} else {

res.end("wrong");

}

})

}

## 6.2用户注册

*/\*添加用户\*/*

function add\_sql(account, pw, name, sex, birth, phone, email, sid, resume, res) {

addsql = "insert into user(user\_account,user\_pw,user\_name,user\_sex,user\_birth,user\_phone,user\_email,user\_score,user\_sid,user\_resume) values(?,?,?,?,?,?,?,0,?,?)";

addsqlparams = [account, pw, name, sex, birth, phone, email, sid, resume];

connection.query(addsql, addsqlparams, function (err, result) {

if (err) {

console.log(err.message);

return;

} else {

res.sendFile(\_\_dirname + "/public/index.html");

}

})

}

## 6.3注册校验

*/\*注册帐号校验\*/*

function check\_sql(res, account) {

sql = "select \* from user where user\_account=?";

sqlparams = [account];

console.log("account:" + account);

connection.query(sql, sqlparams, function (err, result) {

if (err) {

console.log(err.message);

return;

}

console.log("成功进行sql");

var str = false;

for (var a in result) {

str = true;

}

if (str) {

console.log("帐号已被注册");

res.write("帐号已被注册");

res.end();

} else {

res.end();

}

})

}

## 6.4读取用户信息

*/\*读取用户信息\*/*

function read\_user(res, id) {

sql = "select \* from user where user\_id=?";

sqlparams = [id];

connection.query(sql, sqlparams, function (err, result) {

if (err) {

console.log(err.message);

return;

} else {

console.log(JSON.stringify(result));

res.end(JSON.stringify(result));

}

})

}

## 6.5添加收藏

*/\*添加收藏\*/*

function collect(res, sc\_price, commodity\_id, user\_id) {

var t1 = new Date().toLocaleDateString();

var t2 = new Date().toLocaleTimeString();

var t3 = t1 + " " + t2;

sql = "insert into sc(sc\_time,sc\_price,commodity\_id,user\_id) values(?,?,?,?)";

sqlparams = [t3, sc\_price, commodity\_id, user\_id];

connection.query("update commodity set commodity\_scl=commodity\_scl+1 where commodity\_id=" + commodity\_id + "", function (err, result) {

if (err) {

console.log(err.message);

return;

} else {

connection.query(sql, sqlparams, function (err, result) {

if (err) {

console.log(err.message);

return;

} else {

res.write("收藏发布成功");

res.end();

}

})

}

})

}

# 七、商品模块代码

## 7.1关键词搜索

*/\*关键词搜索\*/*

function keysearch(res, key) {

sql = "select commodity\_id,commodity\_picture,commodity\_name,commodity\_price from commodity where commodity\_name like ? or commodity\_key like ?";

sqlparams = ["%" + key + "%", "%" + key + "%"];

connection.query(sql, sqlparams, function (err, result) {

if (err) {

console.log(err.message);

return;

} else {

res.end(JSON.stringify(result));

}

})

}

## 7.2读取商品信息

*/\*读取商品信息\*/*

function read\_commodity(res, id) {

*//console.log(id);*

sql = "select \* from commodity where commodity\_id=?";

sqlparams = [id];

connection.query(sql, sqlparams, function (err, result) {

if (err) {

console.log(err.message);

return;

} else {

*//console.log(JSON.stringify(result));*

res.end(JSON.stringify(result));

}

})

}

## 7.3读取评论信息

*/\*读取评论信息\*/*

function read\_pl(res, id) {

sql = "select \* from pl where commodity\_id=?";

sqlparams = [id];

connection.query(sql, sqlparams, function (err, result) {

if (err) {

console.log(err.message);

return;

} else {

console.log(JSON.stringify(result));

res.end(JSON.stringify(result));

}

})

}

## 7.4上架商品

*/\*上架商品\*/*

function onsale(res, name, picture, price, user\_id, intro, key) {

var t1 = new Date().toLocaleDateString();

var t2 = new Date().toLocaleTimeString();

var t3 = t1 + " " + t2;

console.log(t3);

sql = "insert into commodity(commodity\_name,commodity\_picture,commodity\_price,user\_id,commodity\_intro,commodity\_key,commodity\_scl,commodity\_pll,commodity\_uptime,commodity\_score) values(?,?,?,?,?,?,0,0,?,0)";

sqlparams = [name, picture, price, user\_id, intro, key, t3];

connection.query(sql, sqlparams, function (err, result) {

if (err) {

console.log(err.message);

return;

} else {

res.send("商品上架成功");

}

})

}

## 7.5下架商品

*/\*下架商品\*/*

function downsale(res, id) {

sql = "delete from commodity where id=?";

sqlparams = [id];

connection.query(sql, sqlparams, function (err, result) {

if (err) {

console.log(err.message);

return;

} else {

res.write("商品下架成功");

res.end();

}

})

}

## 7.6发布评论

*/\*发布评论\*/*

function comment(res, commodity\_id, user\_id, pl\_content) {

var t1 = new Date().toLocaleDateString();

var t2 = new Date().toLocaleTimeString();

var t3 = t1 + " " + t2;

sql = "insert into pl(commodity\_id,user\_id,pl\_content,pl\_time) values(?,?,?,?)";

sqlparams = [commodity\_id, user\_id, pl\_content, t3];

connection.query("update commodity set commodity\_pll=commodity\_pll+1 where commodity\_id=" + commodity\_id + "", function (err, result) {

if (err) {

console.log(err.message);

return;

} else {

connection.query(sql, sqlparams, function (err, result) {

if (err) {

console.log(err.message);

return;

} else {

res.write("评论发布成功");

res.end();

}

})

}

})

}

# 八、管理员模块代码

## 8.1删除用户

*/\*删除用户\*/*

function delete\_sql(id, res) {

addsql = "delete from user where user\_id=?";

addsqlparams = [id];

connection.query(addsql, addsqlparams, function (err, result) {

if (err) {

console.log(err.message);

return;

} else {

res.send("删除成功！");

}

})

}

# 九、页面路由跳转

app.get('/', function (req, res) {

res.sendFile(\_\_dirname + "/public/load.html");

});

app.get('/load', function (req, res) {

res.sendFile(\_\_dirname + "/public/index.html");

})

app.get('/post\_data', function (req, res) {

res.sendFile(\_\_dirname + "/public/add.html");

})

app.get('/delete\_data', function (req, res) {

res.sendFile(\_\_dirname + "/public/delete.html");

})

app.get('/homepage', function (req, res) {

res.sendFile(\_\_dirname + "/public/homepage.html");

})

app.get('/onsale', function (req, res) {

res.sendFile(\_\_dirname + "/public/onsale.html");

})

app.get('/comment', function (req, res) {

res.sendFile(\_\_dirname + "/public/comment.html");

})

app.get('/commodity\_index', function (req, res) {

res.sendFile(\_\_dirname + "/public/commodity\_index.html")

})

*/\*添加收藏\*/*

app.get('/collect', function (req, res) {

var arg1 = url.parse(req.url, true).query;

collect(res, arg1.sc\_price, arg1.commodity\_id, arg1.user\_id);

})

*/\*登录\*/*

app.post('/load', urlencode, function (req, res) {

load\_sql(res, req.body.account, req.body.pw);

})

*/\*注册\*/*

app.post("/post\_data", urlencode, function (req, res) {

add\_sql(req.body.account, req.body.pw, req.body.name, req.body.sex, req.body.birth, req.body.phone, req.body.email, req.body.sid, req.body.resume, res);

})

*/\*注册帐号校验\*/*

app.post('/add\_check', function (req, res) {

check\_sql(res, req.body.account);

})

*/\*关键词搜索\*/*

app.post('/keysearch', urlencode, function (req, res) {

keysearch(res, req.body.key);

})

*/\*读取商品信息\*/*

app.get('/readcommodity', function (req, res) {

var arg1 = url.parse(req.url, true).query;

read\_commodity(res, arg1.commodity\_id);

})

*/\*读取用户信息\*/*

app.post('/readuser', urlencode, function (req, res) {

read\_user(res, req.body.user\_id);

})

*/\*读取评论信息\*/*

app.get('/readpl', function (req, res) {

console.log("正在读取");

var arg1 = url.parse(req.url, true).query;

read\_pl(res, arg1.commodity\_id);

})

*/\*上架商品\*/*

app.post('/onsale', multipart, function (req, res) {

console.log(req.files);

var des\_file = \_\_dirname + "/public/user/" + req.files.picture.originalFilename;

var fileaddress = "/user/" + req.files.picture.originalFilename;

console.log(des\_file);

fs.readFile(req.files.picture.path, function (err, data) {

fs.writeFile(des\_file, data, function (err) {

if (err) {

console.log(err);

} else {

onsale(res, req.body.name, fileaddress, req.body.price, req.body.user\_id, req.body.intro, req.body.key);

}

});

});

})

*/\*下架商品\*/*

app.post('/downsale', urlencode, function (req, res) {

downsale(res, req.body.commodity\_id);

})

*/\*发布评论\*/*

app.post('/comment', urlencode, function (req, res) {

comment(res, req.body.commodity\_id, req.body.user\_id, req.body.pl\_content);

})

*/\*删除用户\*/*

app.post("/delete\_data", urlencode, function (req, res) {

delete\_sql(req.body.user\_id);

})

1. [↑](#footnote-ref-1)