

区块链原理与技术 Project 最终报告

软件工程（电子政务） 16340291 张马良

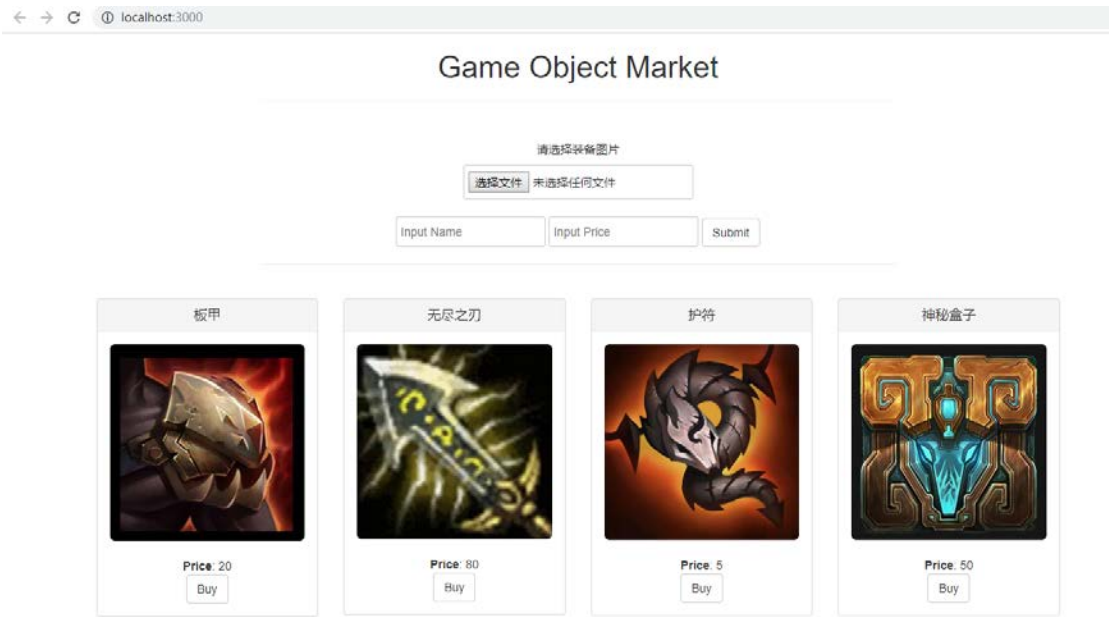
目录

区块链原理与技术 Project 最终报告	1
1. 项目地址	1
2. 项目背景	2
3. 软件构思	2
5. 使用说明	3
6. 使用实例	3
7. 参考资料	9

1. 项目地址

游戏物品交易平台：<https://github.com/PeanutADi/GameObjectMarket>

项目效果图：



项目特点：

可以读取 Json 数据上架商品（符合实际情况），也可以自由的填写表单上架商品（方便测试及平时使用）。

项目准备：

- 安装 Node.JS 环境。下载我的源码。
- 安装 Truffle,并且 solidity 的编译器版本在 0.5.0 以下。

- iii) 在默认的浏览器中安装 MetaMask, 并使用 Truffle 提供的助记词 (candy maple cake sugar pudding cream honey rich smooth crumble sweet treat) 生成的账号。

2. 项目背景

我的项目目标是: 构建基于区块链的游戏物品交易平台。

现在国内的游戏用户很多, 近几年, 我国手机端的网络游戏用户数量增长最为快速, 网页游戏及客户端游戏用户增长较为缓慢, 但是也呈两位数的增长速度。从数据上发现, 2014 年我国网络游戏收入已经破千亿。

不管是卖家有冗余的物品或者亟需回笼资金, 还是买家不想费时费力打装备, 大量的游戏交易需求都是存在的。传统的游戏交易平台主要交易业务包括游戏点卡、游戏虚拟物品、货币、账号交易、游戏代练等。而伴随着传统端游市场逐渐紧缩, 竞技类及道具收费游戏增多, 移动端游戏爆发式增长, 游戏玩家对点卡大幅需求降低, 收费制基本仅剩为数不多、老而弥坚的 MMORPG 游戏在继续沿用。随之而来的还有虚拟物品与虚拟货币市场的减少, 再

加上游戏账号交易市场被官方渠道瓜分。而随着交易市场的逐渐细化, 传统交易平台提供的代练服务已经不能够满足主流市场的代练需求。在 2016 年, 广电总局批准出版国产游戏约 3800 款, 其中客户端游戏仅占 2.0%, 新上市端游的减少, 导致游戏交易平台只能依靠老端游市场维持门面, 很难有新端游交易市场增量, 因此游戏交易平台需要改革, 利用手机游戏市场的巨大红利。

抛开传统的游戏交易平台的困境不谈, 现如今最出色的几个游戏交易平台是什么样的呢? 5173 是传统平台霸主, 然而在移动端所占份额不多, 但也有可观的占有份额; 淘宝凭借流量平台优势, 占据了手游最大的交易份额; 阿里系的交易猫则凭借阿里和 UC 的渠道优

势, 以及较为优惠的平台中介费占据了除了淘宝外的大头; 网易系的游戏则主要在网易官方推出的藏宝阁进行交易。由此可见, 用户在交易平台的选择上看重的地方主要有以下几点:

1. 平台的公信力: 主流的交易平台都有极为强大的信用背书, 如 5173 是靠多年经营的口碑, 淘宝则是国民网购时想到的第一电商品牌, 交易猫和藏宝阁分别背靠阿里和网易两大企业。用户肯定不想让自己的钱打水漂, 交易平台的公信力是很重要的。
 2. 中介费用: 卖家不想因为过高的中介费用而提高自己物品的售价使得买家望而却步, 买家不想因为中介费用而付出额外的钱。比如交易猫凭借免中介费吸引大量用户, 淘宝更是不存在“中介费”一说。中介费甚至可能比公信力在某些用户心中有着更重要的影响力, 不少用户因为贪图便宜的中介费用选择了小的平台甚至私下交易, 被骗的也不在少数。
 3. 物品的展示方法: 如果只用图片展示, 有可能导致商品信息展示不全或者卖家恶意造假, 因此对于商品交易平台应有明确的独一无二的记录方式使得买家卖家信息对等。
- 以上就是我对游戏交易平台现状的分析。

3. 软件构思

我认为区块链能完美解决游戏物品交易平台的需求:

1. 公信力方面, 去中心化的特点使得搭建在区块链的游戏交易平台无需信用背书就很难发生欺诈行为。
2. 中介费用几乎不存在, 当交易成交时收取少量 token 即可, 更像是收税而非中介费,

费率较低。

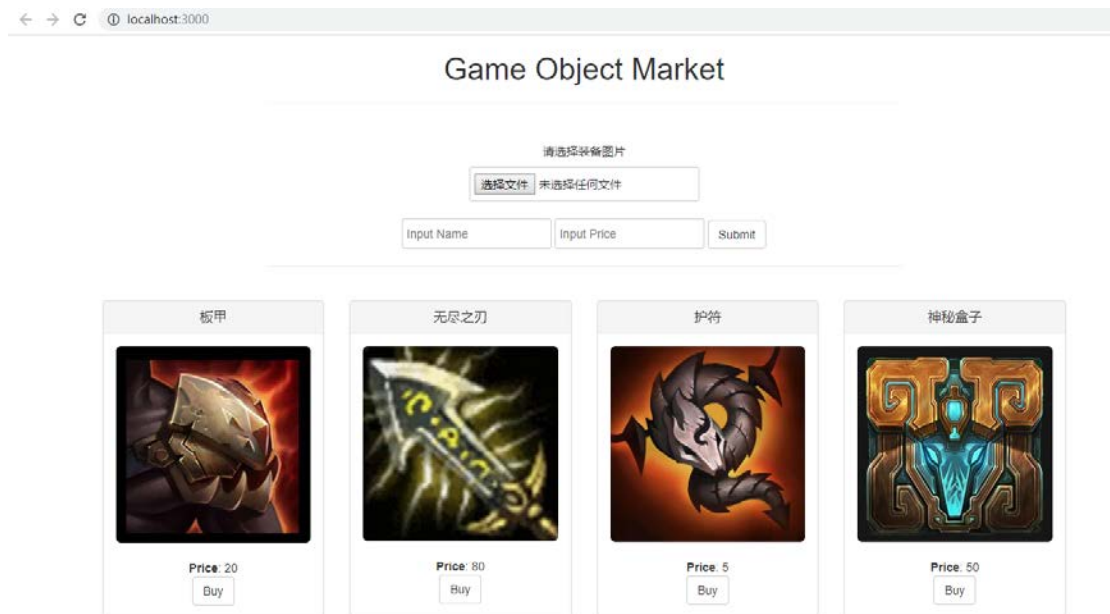
3. 物品或账号及每笔交易都有独特的 hash 值，卖家无法做出欺诈行为，若卖家试图欺诈（如描述不符）很容易就能被发觉。

5. 使用说明


1. 本项目需要运行在 windows 环境，及 Solidity 版本需要在 0.5.0 版本以下。
2. 下载源代码后，删除已有的编译文件 build 文件夹。
3. 打开 cmd，进入项目目录，输入 truffle develop 进入项目控制台。输入 compile 编译。输入 migrate 部署。
4. 新开 cmd，进入项目目录，输入 npm run dev，打开 lite-server，在浏览器中打开项目。
5. 在项目的可视化窗口中，文件的选择窗口是用来选择物品的图片。在输入完物品的价格、名称后，点击 submit 将商品提交到市场中。
6. 对于市场中的商品，单击 Buy 进行购买，在经过钱包的确认后购买成功。

6. 使用实例

在使用中，为了演示方便，我没有对市场进行初始化，即最初市场没有商品！正常的市场打开的样子应该是：



进入项目目录，输入 Truffle develop 进入 truffle 控制台，以便进行编译和部署操作。

 选择命令提示符 - truffle.cmd develop

```
F:\FinalMarket>truffle.cmd develop
Truffle Develop started at http://127.0.0.1:9545/

Accounts:
(0) 0x627306090abab3a6e1400e9345bc60c78a8bef57
(1) 0xf17f52151ebef6c7334fad080c5704d77216b732
(2) 0xc5fdf4076b8f3a5357c5e395ab970b5b54098fef
(3) 0x821aea9a577a9b44299b9c15c88cf3087f3b5544
(4) 0x0d1d4e623d10f9fba5db95830f7d3839406c6af2
(5) 0x2932b7a2355d6fecc4b5c0b6bd44cc31df247a2e
(6) 0x2191ef87e392377ec08e7c08eb105ef5448eced5
(7) 0x0f4f2ac550a1b4e2280d04c21cea7ebd822934b5
(8) 0x6330a553fc93768f612722bb8c2ec78ac90b3bbc
(9) 0x5aeda56215b167893e80b4fe645ba6d5bab767de

Private Keys:
(0) c87509a1c067bbde78beb793e6fa76530b6382a4c0241e5e4a9ec0a0f44dc0d3
(1) ae6ae8e5ccbfb04590405997ee2d52d2b330726137b875053c36d94e974d162f
(2) 0dbbe8e4ae425a6d2687f1a7e3ba17bc98c673636790f1b8ad91193c05875ef1
(3) c88b703fb08cbea894b6aeff5a544fb92e78a18e19814cd85da83b71f772aa6c
(4) 388c684f0ba1ef5017716adb5d21a053ea8e90277d0868337519f97bede61418
(5) 659cbb0e2411a44db63778987b1e22153c086a95eb6b18bdf89de078917abc63
(6) 82d052c865f5763aad42add438569276c00d3d88a2d062d36b2bae914d58b8c8
(7) aa3680d5d48a8283413f7a108367c7299ca73f553735860a87b08f39395618b7
(8) 0f62d96d6675f32685b5b5b8ac13cda7c23436f63efbb9d07700d8669ff12b7c4
(9) 8d5366123cb560bb606379f90a0bfd4769eccc0557f1b362dcae9012b548b1e5

Mnemonic: candy maple cake sugar pudding cream honey rich smooth crumble sweet treat
⬢ Important ⬢ : This mnemonic was created for you by Truffle. It is not secure.
Ensure you do not use it on production blockchains, or else you risk losing funds.

truffle(develop)>
```

在 truffle develop 下，输入 compile 进行编译：

```
truffle(develop)> compile
```

当出现将 abi 写入文件夹时，编译成功：

```
Writing artifacts to .\build\contracts
```

接着将他们部署到我的私链上：

```
truffle(develop)> migrate
Using network 'develop'.

Running migration: 1_initial_migration.js
  Deploying Migrations...
  ... 0x59980d7f6addaa294c0f069e9852a5fd0109b310f3c700638e02d4cd962040e0
  Migrations: 0x8cdaf0cd259887258bc13a92c0a6da92698644c0
Saving successful migration to network...
  ... 0xd7bc86d31bee32fa3988f1c1eabce403a1b5d570340a3a9cdba53a472ee8c956
Saving artifacts...
Running migration: 2_deploy_contracts.js
  Deploying Adoption...
  ... 0x2f8126fd349335015ecee7b805da74619bb91b5c9112a6e65c14957fafa70eac
  Adoption: 0x345ca3e014aaf5dca488057592ee47305d9b3e10
Saving successful migration to network...
  ... 0xf36163615f41ef7ed8f4a8f192149a0bf633fela2398ce001bf44c43dc7bdda0
Saving artifacts...
```

接着，在新开的 cmd 中进入目录下并输入 npm run dev 命令，会启动我为项目配置的 lite-server 服务器，即图形化界面。

```
ca: lite-server
F:\>cd FinalMarket
F:\FinalMarket>npm run dev
> pet-shop@1.0.0 dev F:\FinalMarket
> lite-server

** browser-sync config **
{ injectChanges: false,
  files: [ './**/*.html', './**/*.css', './**/*.js' ],
  watchOptions: { ignored: 'node_modules' },
  server:
    { baseDir: [ './src', './build/contracts' ],
      middleware: [ [Function], [Function] ] } }
[Browsersync] Access URLs:
-----
Local: http://localhost:3000
-----
UI: http://localhost:3001
-----
[Browsersync] Serving files from: ./src
[Browsersync] Serving files from: ./build/contracts
[Browsersync] Watching files...
18.12.28 21:14:04 200 GET /index.html
18.12.28 21:14:05 200 GET /css/bootstrap.min.css
18.12.28 21:14:05 200 GET /js/bootstrap.min.js
18.12.28 21:14:05 200 GET /js/web3.min.js
18.12.28 21:14:05 200 GET /js/truffle-contract.js
18.12.28 21:14:05 200 GET /js/app.js
```

Game Object Market 界面在浏览器中打开：

Game Object Market

请选择装备图片

选择文件

未选择任何文件

Input Name

Input Price

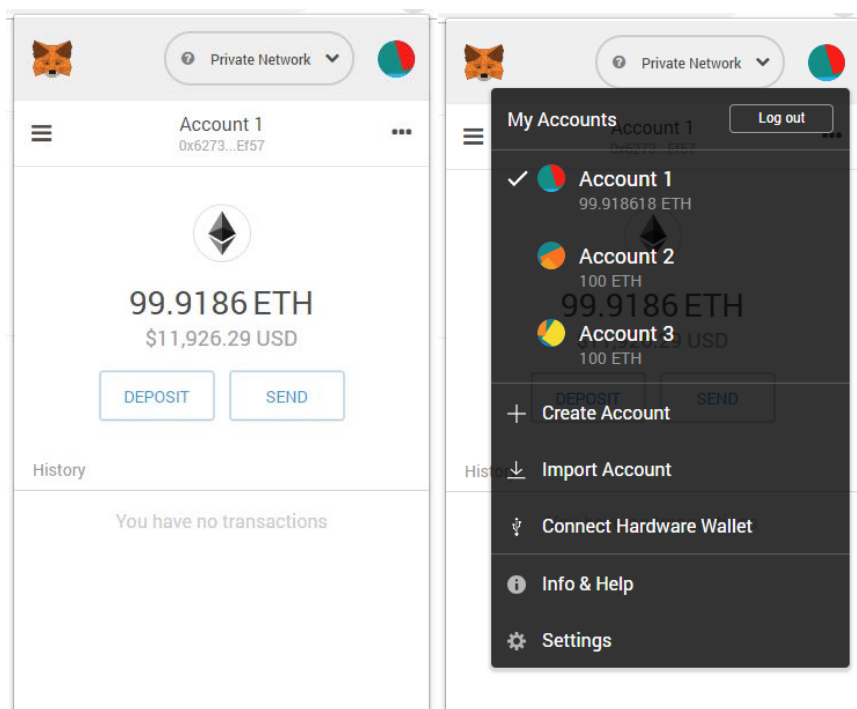
Submit

因为我没有为它初始化设定内容，所以页面是空的。

页面的上半部分用于卖家上架货物，下半部分将展示已经上架的货物。

我安装了 MetaMask 作为我的钱包进行测试，并且准备了三个账户。第一个账户在部署时消耗了少量的 gas，因此余额不是 100。

这三个账户是用 truffle 的助记词初始化的，因此地址与网络上同样使用 truffle 的账户一样。



首先测试上架功能：

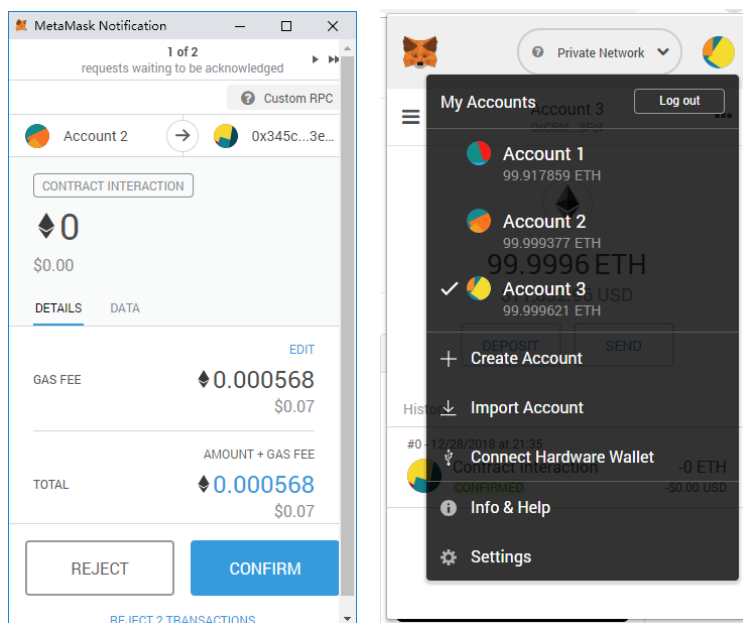
在上架区使用账户二如下输入：（其中，图片是随意选择的）

请选择装备图片

选择文件 wujin.jpg

无尽之刃 2 Submit

单击提交后，因为上传物品要调用函数，保存账户的地址，所以会消耗 gas，MetaMask 会弹出页面提醒确认。单击 Confirm，页面会刷新出现上架的物品：



左图：上架货物需要确认


右图：上架多个物品后，每个账户都消耗了 gas，因此余额都不是 100ETH 了

Game Object Market

请选择装备图片

wujin.jpg

无尽之刃







Price: 2

在钱包中，切换不同的账户，并上架多个物品：

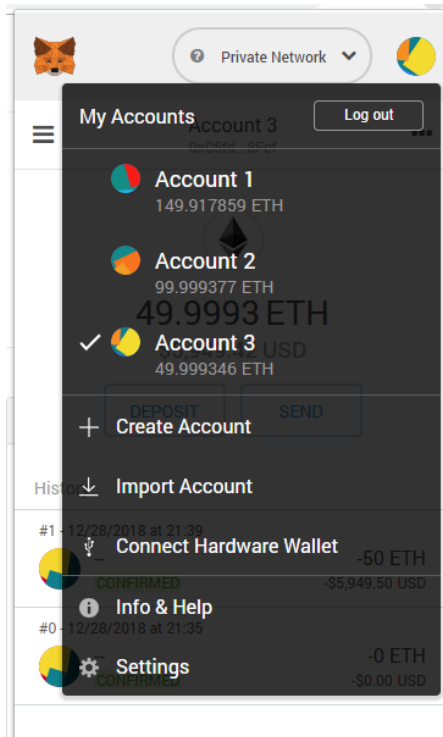
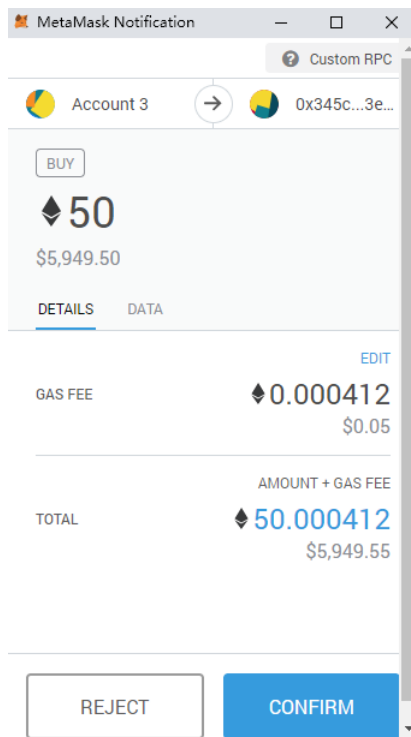
Game Object Market

请选择装备图片

banjia.jpg

<p>无尽之刃</p>  <p>Price: 2</p> <p><input type="button" value="Buy"/></p>	<p>猎人护符</p>  <p>Price: 5</p> <p><input type="button" value="Buy"/></p>	<p>神秘盒子</p>  <p>Price: 50</p> <p><input type="button" value="Buy"/></p>	<p>盔甲</p>  <p>Price: 30</p> <p><input type="button" value="Buy"/></p>
---	---	---	--

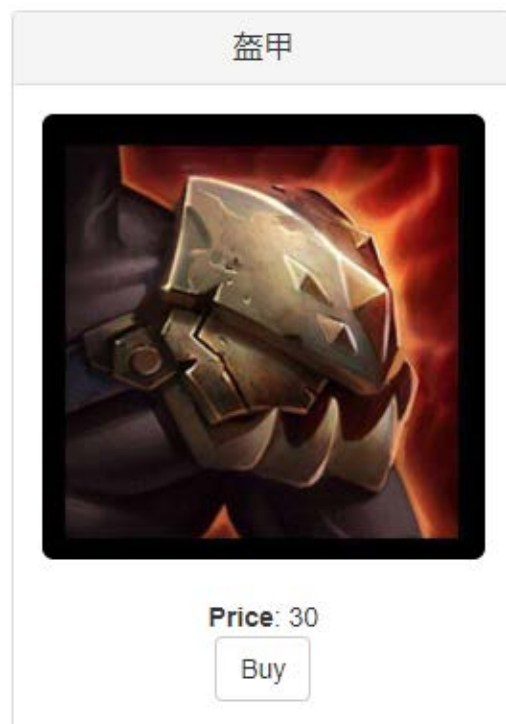
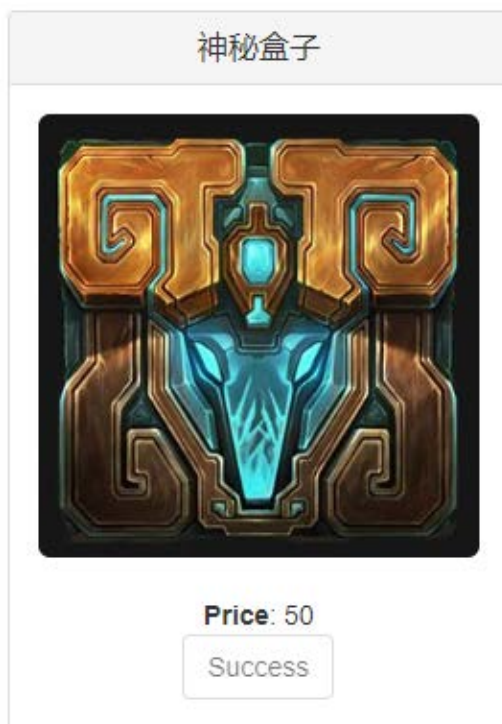
之后使用账户三向账户一上架的某个商品进行购买：



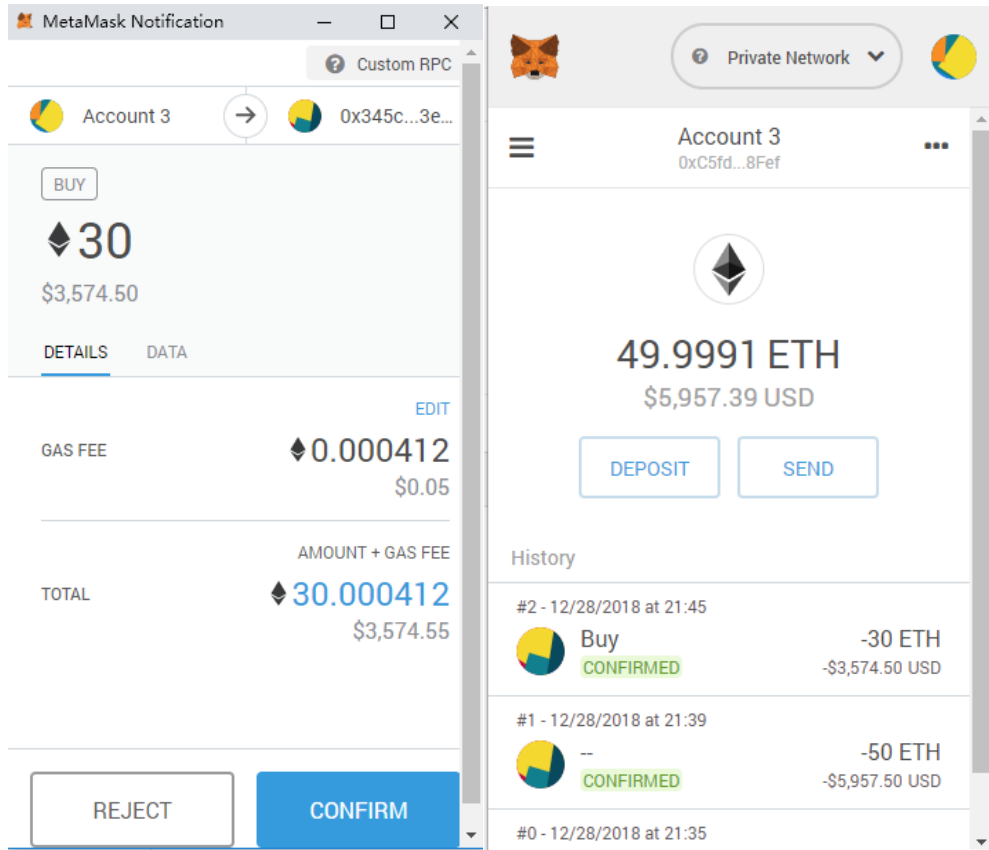
左图：购买时需要进行确认

右图：购买后各个账户的余额

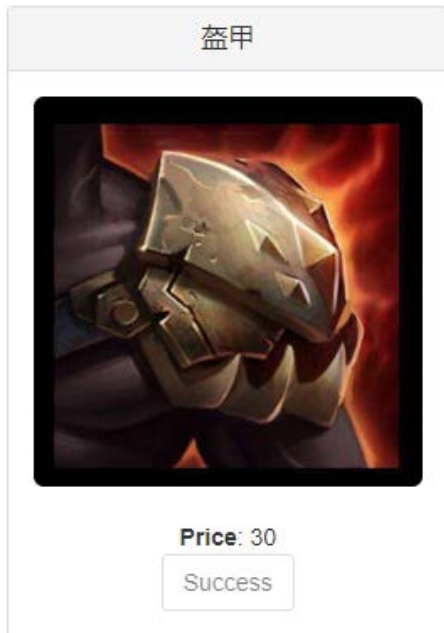
因为涉及交易，MetaMask 弹出了确认框。单击确认后，即买下了这个商品。查看账户的余额，上架该商品的账户一增加了 50ETH，而买下该商品的账户三减少了 50ETH。并且买下的商品的购买按钮会变得不可用。



而假如账户三想购买自己上架的商品，也会弹出确认界面，并且会消耗 Gas 进行购买（下左图）。在确认后，虽然账户的 History 中记录了这笔交易，但由于自己给自己转账所以余额并未增加，只是白白消耗了 Gas，避免了用户无意义的对商品进行上架、下架。



虽然是购买自己的商品，但也是购买了并且成功，因此商品也应该下架，即将购买按钮设置为不可用：



7. 参考资料

官方例子 petshop: <http://truffleframework.com/tutorials/pet-shop>