

玩区块链游戏迷恋猫 CryptoKitties, 学习 区块链技术赚 ETH

便于读者更清晰阅读本文，先列出本文的内容大纲：

玩区块链游戏迷恋猫 CryptoKitties, 学习区块链技术赚 ETH

以太坊开发技术基础

以太坊概述

Solidity

ERC-20 和 ERC-721

搭建以太坊全节点

迷恋猫系统结构

迷恋猫 DApp

迷恋猫智能合约

迷恋猫游戏规则

以太坊钱包就是用户的 ID

买猫

卖猫和出租种猫

出租种猫

生猫

向他人赠送猫咪

使用 web3js 读写区块链上的迷恋猫数据

库依赖

初始化

读取猫的代数和休息时间 cooldown_index

读取猫的拍卖信息

获取当前猫的总数量

通过程序买猫

通过程序卖猫

通过程序生猫

检查新挂单的猫咪

使用 api.cryptokitties.co 读取数据

通过程序赚 ETH

低买高卖

批量生猫抢靓号

选择合适的基因生出高价的猫

当猫的接生婆赚 ETH

以太坊开发技术基础

以太坊概述

以太坊是可编程的区块链，是业内公认的区块链 2.0 代表项目。可以将以太坊理解为一个操作系统，使用 Solidity 等语言编写智能合约发布应用到链上，使用 Go、Java、Python、JavaScript 等语言在链下调用链上的智能合约读写区块链数据，通过这种方式实现各种各样的区块链应用。

比特币的总上限是2100万，而以太坊的内置代币以太币（Ether）没有确切的总量上限。目前以太坊大概每15秒出一个新块，一个新块奖励矿工 3 ETH。以太坊的设计者认为随着时间流逝总会发生因为粗心和死亡等原因带来的币的遗失，假设币的遗失是每年货币供应量的一个固定比例，则最终总的流通中的货币供应量会稳定在一个等于年货币发行量除以遗失率的值上，使得供应量会趋于稳定。

比特币缺少图灵完备性，尽管比特币脚本语言可以支持多种计算，但是它不能支持所有的计算，如不支持 for 循环。以太坊是准图灵完备的，之所以增加“准”，是因为智能合约在以太坊区块链上执行时是受限的。在以太坊区块链上执行交易（转账、调用智能合约）需要消耗 Gas，一般来说操作步骤越复杂需要的 Gas 越多，而一个块有 Gas 上限（目前约为 800 万）。向普通账户做1次转账操作目前消耗 2.1 万 Gas。在块 Gas 上限为 800 万时，假设调用一个智能合约中某个函数时会向400个账户转账，因为会至少消耗 $400 * 2.1 \text{ 万} = 820 \text{ 万 Gas}$ ，超出块的 Gas 上限 800 万，合约调用会失败。

关于以太坊更详细的介绍可以参考：[以太坊白皮书](#)和[以太坊黄皮书](#)。

Solidity

Solidity 是一种在语法上类似 JavaScript 的高级语言，编译 Solidity 代码可以生成以太坊虚拟机代码。Solidity 是静态类型语言，支持继承、库和复杂的用户定义类型等特性。使用 Solidity 可以很容易创建投票、众筹、封闭拍卖、多重签名钱包、迷恋猫游戏之类的智能合约。

由于以太坊区块链的限制，在链上无法读取链下数据，使用 Solidity 你也无法来调用传统的 API，例如你无法调用某天气网站提供的天气 API。另外在以太坊区块链上，无法让程序在指定时间自动运行。迷恋猫游戏为了真实性，在猫怀孕后不是立即生出小猫，而是需要在满足一定条件后由外部来触发生猫函数为猫提供接生服务，一些开发者根据游戏的这个特性还可以赚猫的接生费，后文会有详细说明。

[etherscan.io](#) 提供了验证程序源码的服务。原理是使用公开的代码及指定的编译器版本再编译一次程序，然后和发布到区块链的以太坊的二进制代码做比对，如果一致说明公开

的代码就是在区块链上运行的那份代码。下图是一份通过验证的代码截图。

The screenshot shows a web interface with tabs: Transactions, Token Transfers, Contract Source (highlighted with a red box and 'Yes'), Read Smart Contract, and Comments. A warning message is displayed: 'Warning: The compiled contract might be susceptible to ZeroFunctionSelector (very low-severity), DelegateCallReturnValue SkipEmptyStringLiteral (low-severity) Solidity compiler bugs.' Below this, a green checkmark indicates 'Contract Source Code Verified'. A table shows contract details: Contract Name: DSToken, Compiler Version: v0.4.11+commit.68ef5810, Optimization Enabled, and Runs (Optimiser). The Contract Source Code is displayed in a text area, showing Solidity code for an ERC20 token. The code includes functions like totalSupply, balanceOf, allowance, transfer, transferFrom, and approve, as well as events Transfer and Approval. A red box highlights the 'contract ERC20' line. A large 'GitChat' watermark is overlaid on the code.

```
261 }
262 }
263
264 - contract ERC20 {
265     function totalSupply() constant returns (uint supply);
266     function balanceOf( address who ) constant returns (uint value);
267     function allowance( address owner, address spender ) constant returns (uint _allowance);
268
269     function transfer( address to, uint value) returns (bool ok);
270     function transferFrom( address from, address to, uint value) returns (bool ok);
271     function approve( address spender, uint value ) returns (bool ok);
272
273     event Transfer( address indexed from, address indexed to, uint value);
274     event Approval( address indexed owner, address indexed spender, uint value);
275 }
276
277 - contract DSTokenBase is ERC20, DSMath {
278     uint256 _supply;
279     mapping (address => uint256) _balances;
280     mapping (address => mapping (address => uint256)) _approvals;
```

可以访问[这里](#)，进一步了解 Solidity。

ERC-20 和 ERC-721

ERC-20 和 ERC-721 都是以太坊 EIP（Ethereum Improvement Proposal，以太坊改进协议），更多 EIP 见[这里](#)。

ERC-20 定义了一份代币标准，可以理解为定义了一个接口类，在实现具体的 ERC-20 代币时，要给出各个接口的具体实现，如获取代币名称、代币符号、总供应量、小数位数、转账等。使用 ERC-20，可以大幅度降低发币成本，发币方无需开发钱包和区块链浏览器，交易所也可以轻松支持新的代币充值提现等操作。

ERC-20 币是同质的，你的一个币和我的一个币是等价的。ERC-721 是非同质代币（Non-fungible Tokens），每个 ERC-721 有唯一的 ID，转账时，不再是转多少币，而是转某个 tokenId，如 transferFrom：

```
function transferFrom(address _from, address _to, uint256
_tokenId) external payable;
```

迷恋猫实现了 ERC-721 规范，每个猫有一个唯一的 ID，玩家花 ETH 买猫时，智能合约会调用 transferFrom 修改猫的所有者。

搭建以太坊全节点

我使用了以太坊 Go 客户端搭建的全节点，参考文档见[这里](#)，几个注意事项如下：

- 电脑配置不能太低。我刚开始使用的是阿里云1核 CPU、2500 MHz 的 ECS，发现怎么也同步不到最新块，升级到了4核后同步正常了；
- 第一次同步时使用 `--fast` 选项，可以更快地同步到最新块，目前（2018-04-02）使用 `--fast` 选项同步到最新块后预计占用 60G 空间；
- geth 运行时间长了可能会有问题，我使用守护进程启动 geth，每 4 个小时 kill 掉 geth，目前基本上能一直同步到最新块；
- 硬盘空间要足够大，建议至少1T以上。我3个月前使用 `--fast` 同步完成后才占 40 多 G 空间，之后正常模式同步后硬盘占用空间快速增长，运行3个月左右已经 500G 了。

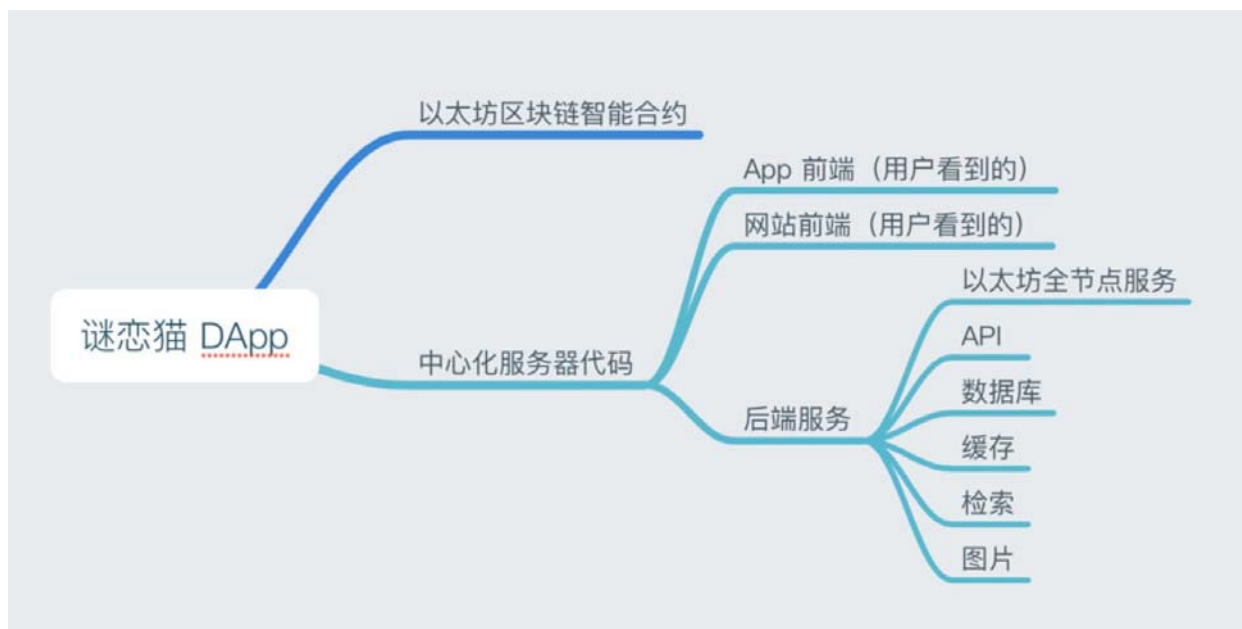
为了方便、更快速的调用相关 API，建议在本地服务器上搭建一个以太坊全节点，并保持同步到最新区块高度。如果想通过程序买卖猫但又不想自己搭建全节点，可以使用 MetaMask 的节点，API 地址请访问[这里](#)，使用 MetaMask 的节点 API，监听事件可能会受到影响，一个方法是遍历某个区块的所有交易信息，然后选择自己关注的交易信息再做相关处理。

迷恋猫系统结构

如果你还没有在 www.cryptokitties.co 买过猫，建议先买只猫后再阅读后续内容。一些入门攻略见[这里](#)，遇到问题可通过迷恋猫 QQ 群（728507998）咨询。

迷恋猫 DApp

迷恋猫是一个 DApp（Decentralized Application），部分程序部署在区块链上，部分程序部署在中心化的服务器上，总体架构如下图：



迷恋猫智能合约

迷恋猫在以太坊区块链上一共有4个智能合约：

- [CryptoKittiesCore](#)：核心代码，已开源，2016行代码；
- [CryptoKittiesSalesAuction](#)：猫拍卖机制，已开源，604行代码；
- [CryptoKittiesSiringAuction](#)：配种拍卖机制，已开源，589行代码；
- [geneScience](#)：基因工程，未开源。

以玩家挂单卖猫为例，说明相关步骤：

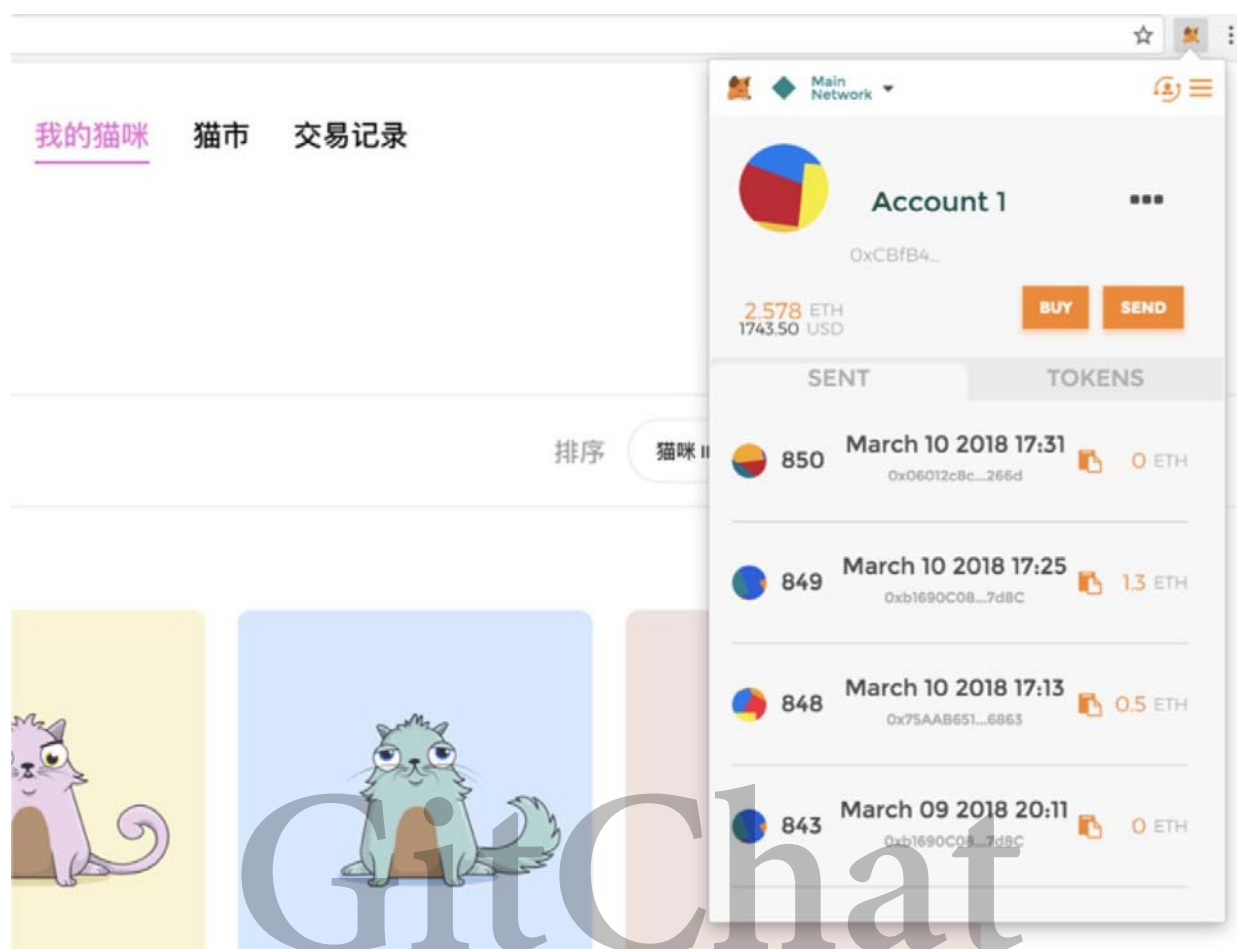
1. 玩家在自己某只猫的页面点击出售按钮，设置出售价格，通过 MetaMask 钱包操作后等待（页面上猫的状态是非在售；MetaMask 交易状态是 pending 中）；
2. MetaMask 将交易广播出去（页面上猫的状态是非在售；MetaMask 交易状态是 pending 中）；
3. 交易被矿工打包到区块中（页面上猫的状态是非在售；MetaMask 交易状态是 pending 中）；
4. 迷恋猫的以太坊节点监听到卖猫事件，更新数据库中猫的信息、更新缓存数据、更新搜索索引数据；MetaMask 的以太坊节点更新交易状态（页面上猫的状态是在售；MetaMask 交易状态是已完成）；

用户卖猫后，在页面上不是立即可以看到猫处于在售中，而是需要等待一会儿才能看到，整个过程是异步的。这也为懂技术的玩家留下机会，可以在猫刚开卖还未在页面上显示出售中就能立即买下猫，详细方法见后文。

迷恋猫游戏规则

以太坊钱包就是用户的 ID

一般涉及到用户数据的网站或 App，都会要用户注册帐号，哪怕是使用第三方授权。而对于迷恋猫游戏，即使你之前从未访问过网站，只要有你的以太坊钱包地址，其他人就可以给你送猫。



安装 [MetaMask](#) 插件后第一次登陆网站，需要你按提示点击一个消息签名按钮，以便验证当前用户身份。原理是椭圆曲线加密算法，相关函数为 `web3.eth.sign(address, dataToSign)`、`ecRecover`，具体用法可以搜索下。

买猫

在[迷恋猫](#)页面，点击顶部导航“猫市”，默认就是待收养的猫猫。点击“筛选猫咪”可以通过多种方式做筛选，找到自己的目标猫咪。初期玩建议按价格从低到高排序，选两三只便宜的猫咪。搜索功能并不是直接从区块链读取数据的，而是通过同步区块链数据后在中心化服务器中建立的索引。

Q 搜索

待收养

待交配

零代猫

所有猫咪

排序 价格 由低到高

125,450 猫咪

猫咪种类

普通猫咪

奇异猫咪

专供猫咪

代数

所有

休息时间

神速

急速

快速

中速

慢速


迟缓

龟速

昏迷


筛选猫咪

等待领养 ≈ 0.002




猫咪 547756 · 8 代 · 中速
♡ 1

等待领养 ≈ 0.002




猫咪 576322 · 7 代 · 中速
♡ 1

等待领养 ≈ 0.002



猫咪 586450 · 9 代 · 迟缓
♡ 2

等待领养 ≈ 0.002



猫咪 586963 · 10 代 · 慢速
♡ 2

点击一只猫咪后，进入单个猫咪页面，再点击“立即购买”就可以买猫了。

猫咪 #553863

猫咪 553863 · 5 代 · 急速 休息时间 ⓘ

Derek#7051 / derek.kitties@gmail.com

主人



♡ 贊 0

即時購買價格
 ≈ 0.0036

剩余時間
4 天

立刻購買



在点击“立即购买”按钮时，会调用 web3js，触发弹出 MetaMask 插件窗口。MetaMask 插件中会显示 Amount（转账额度）、GasLimit（燃料上限）、GasPrice（燃料价格）等参数，部分数据做了隐藏，如购买猫调用的是 bid(uint256_tokenId) 函数。点击“submit”按钮后，MetaMask 会将交易数据提交到以太坊网络，等待矿工打包确认，当矿工将交易数据打包到某个区块中才算真正完成相关函数执行。但成功打包到区块中不一定能成功买到猫，因为交易过程是异步的，在这个过程中也可能有其他人购买同一只猫，如果他人的交易比你的交易优先被矿工打包，你就买不到猫了。另外，卖家也可以取消卖猫。

MetaMask Notification

CONFIRM TRANSACTION

Main Network

Account 1

CBfB42...4a58

7.110 ETH

2715.51 USD

>

b1690C...7d8C

Amount

0.003635 ETH

1.39 USD

Gas Limit

135963

UNITS

Gas Price

2.6

GWEI

Max Transaction Fee

0.000353 ETH

0.13 USD

Max Total

0.003989 ETH

1.52 USD

GitChat

Data included: 36 bytes

RESET

SUBMIT

REJECT

在此简单说下以太坊的 Gas 机制，在以太坊中涉及到任何写数据的操作都有消耗 Gas，矿工打包一般优先打包 gasPrice 高的交易。gas * gasPrice 表示在操作过程中消耗的以太币，剩余的以太币 (gasLimit - gas) * gasPrice 会返回给用户。一般来做操作越复杂，消耗的 Gas 越多，占用的存储空间越大，消耗的 Gas 越多。

卖猫和出租种猫

你买卖达成后后，平台会收取交易的 3.75% 作为手续费。因以太坊区块链上的数据对所有可见，卖猫不支持暗拍（备注：[ENS](#)支持安排，但相对复杂，不合适面向普通用户）。为了让买家以相对和合适的价格找到相对合适的卖家，卖猫时，可以设置起始价、结束价、价格变化时间段（出售期限），价格变化时间段结束后，还会处于出售状态，价格保持为结束价。

🐾 出售猫咪

猫咪 #65807 将被放到公共市场待人领养。猫咪出售价格可在选定时间内上涨或下跌，出售时间结束后价格会保持稳定。隻有取消銷售才能將猫咪從貓市上收回。

起始价	≡	0.01	▲ ▼
结束价	≡	0.005	▲ ▼
出售期限	天数	2	▲ ▼
<div>完成</div>			

当前价格的计算方法:

```
/// @dev Computes the current price of an auction. Factored
out
/// from _currentPrice so we can run extensive unit tests.
/// When testing, make this function public and turn on
/// `Current price computation` test suite.
function _computeCurrentPrice(
    uint256 _startingPrice,
    uint256 _endingPrice,
    uint256 _duration,
    uint256 _secondsPassed
)
    internal
    pure
    returns (uint256)
{
    // NOTE: We don't use SafeMath (or similar) in this
function because
    // all of our public functions carefully cap the maximum
values for
    // time (at 64-bits) and currency (at 128-bits).
_duration is
    // also known to be non-zero (see the require()
statement in
    // _addAuction())
```

```

        if (_secondsPassed >= _duration) {
            // We've reached the end of the dynamic pricing
            portion
            // of the auction, just return the end price.
            return _endingPrice;
        } else {
            // Starting price can be higher than ending price
            (and often is!), so
            // this delta can be negative.
            int256 totalPriceChange = int256(_endingPrice) -
            int256(_startingPrice);

            // This multiplication can't overflow, _secondsPassed
            will easily fit within
            // 64-bits, and totalPriceChange will easily fit
            within 128-bits, their product
            // will always fit within 256-bits.
            int256 currentPriceChange = totalPriceChange *
            int256(_secondsPassed) / int256(_duration);

            // currentPriceChange can be negative, but if so,
            will have a magnitude
            // less than _startingPrice. Thus, this result will
            always end up positive.
            int256 currentPrice = int256(_startingPrice) +
            currentPriceChange;
            return uint256(currentPrice);
        }
    }
}

```

出租种猫

出租种猫和卖猫类似，可以通过出租种猫赚钱，但所有权还是原主人的。相关合约见 [CryptoKittiesSiringAuction](#)。

生猫

生猫规则如下：

- 任意猫都可以充当爸爸或妈妈的角色；
- 交配时不能乱伦；
- 每生育一次，恢复时间变长，直到需要7天时间恢复；
- 孕期 = 怀孕后妈妈的恢复时间，想尽快生出小猫的话，应该使用休息时间短的猫做妈妈；
- 小猫代数 = $\max(\text{爸爸的代数}, \text{妈妈的代数}) + 1$ ；
- 小猫恢复时间 $\text{cooldown_index} = \min(\text{小猫代数} / 2, 13)$ ；

- 不同 cooldownw_index 对应的时间见下面内容。

生猫规则 恢复时间

cooldown_index	恢复时间	cooldown_index	恢复时间
0	1 分钟（神速）	7	4小时（慢速）
1	2 分钟（急速）	8	8小时（慢速）
2	5分钟（急速）	9	16小时（迟缓）
3	10分钟（快速）	10	1天（迟缓）
4	30分钟（快速）	11	2天（龟速）
5	1小时（中速）	12	4天（龟速）
6	2小时（中速）	13	1周（昏迷）

生猫函数如下：

```
/// @dev An internal method that creates a new kitty and
stores it. This
/// method doesn't do any checking and should only be called
when the
/// input data is known to be valid. Will generate both a
Birth event
/// and a Transfer event.
/// @param _matronId The kitty ID of the matron of this cat
(zero for gen0)
/// @param _sireId The kitty ID of the sire of this cat (zero
for gen0)
/// @param _generation The generation number of this cat,
must be computed by caller.
/// @param _genes The kitty's genetic code.
/// @param _owner The initial owner of this cat, must be non-
zero (except for the unKitty, ID 0)
function _createKitty(
    uint256 _matronId,
    uint256 _sireId,
    uint256 _generation,
    uint256 _genes,
```

```

        address _owner)
    internal
    returns (uint)
    {
        // These requires are not strictly necessary, our calling
code should make
        // sure that these conditions are never broken. However!
_createKitty() is already
        // an expensive call (for storage), and it doesn't hurt
to be especially careful
        // to ensure our data structures are always valid.
        require(_matronId == uint256(uint32(_matronId)));
        require(_sireId == uint256(uint32(_sireId)));
        require(_generation == uint256(uint16(_generation)));

        // New kitty starts with the same cooldown as parent
gen/2
        uint16 cooldownIndex = uint16(_generation / 2);
        if (cooldownIndex > 13) {
            cooldownIndex = 13;
        }

        Kitty memory _kitty = Kitty({
            genes: _genes,
            birthTime: uint64(now),
            cooldownEndBlock: 0,
            matronId: uint32(_matronId),
            sireId: uint32(_sireId),
            siringWithId: 0,
            cooldownIndex: cooldownIndex,
            generation: uint16(_generation)
        });
        uint256 newKittenId = kitties.push(_kitty) - 1;

        // It's probably never going to happen, 4 billion cats is
A LOT, but
        // let's just be 100% sure we never let this happen.
        require(newKittenId == uint256(uint32(newKittenId)));

        // emit the birth event
        Birth(
            _owner,
            newKittenId,
            uint256(_kitty.matronId),
            uint256(_kitty.sireId),
            _kitty.genes
        );

        // This will assign ownership, and also emit the Transfer
event as
        // per ERC721 draft
        _transfer(0, _owner, newKittenId);

```

```

        return newKittenId;
    }

```

向他人赠送猫咪

相关智能合约代码如下，transfer 是 ERC-721 规范中的一个接口：

```

    /// @notice Transfers a Kitty to another address. If
    transferring to a smart
    /// contract be VERY CAREFUL to ensure that it is aware of
    ERC-721 (or
    /// CryptoKitties specifically) or your Kitty may be lost
    forever. Seriously.
    /// @param _to The address of the recipient, can be a user or
    contract.
    /// @param _tokenId The ID of the Kitty to transfer.
    /// @dev Required for ERC-721 compliance.
    function transfer(
        address _to, // 猫的接受人
        uint256 _tokenId /* 猫 id */ )
    external
    whenNotPaused
    {
        // Safety check to prevent against an unexpected 0x0
        default.
        require(_to != address(0));
        // Disallow transfers to this contract to prevent
        accidental misuse.
        // The contract should never own any kitties (except very
        briefly
        // after a gen0 cat is created and before it goes on
        auction).
        require(_to != address(this));
        // Disallow transfers to the auction contracts to prevent
        accidental
        // misuse. Auction contracts should only take ownership
        of kitties
        // through the allow + transferFrom flow.
        require(_to != address(saleAuction));
        require(_to != address(siringAuction));

        // You can only send your own cat.
        require(_owns(msg.sender, _tokenId));

        // Reassign ownership, clear pending approvals, emit
        Transfer event.
        _transfer(msg.sender, _to, _tokenId);
    }

```

使用 web3js 读写区块链上的迷恋猫数据

库依赖

我使用的 web3js 0.2x 版本，初始化环境见[这里](#)。需要注意的是 web3js 1.0 版本和 0.2x 版本差别较大，并不兼容。

我的 package.json 内容如下：

```
{
  "name": "kitty_statistics",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "dependencies": {
    "express": "^4.16.2",
    "http-server": "^0.10.0",
    "web3": "^0.20.3"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

初始化

直接贴代码片段，解读见注释：

```
var express = require('express');
var Web3 = require("web3");
var app = express();

// 如果自己没节点，可以使用第三方的服务，但在监听事件时可能无法使用
// var httpProviderAddr = "https://mainnet.infura.io/metamask";

// 需要替换为自己的节点rpc服务的IP和端口号
var httpProviderAddr = "http://127.0.0.1:8545";

console.log('web3 httpProviderAddr: %s', httpProviderAddr);

var web3 = new Web3(new
Web3.providers.HttpProvider(httpProviderAddr));
```


// ABI 数据可以从智能合约链接中拷贝

```
var cryptoKittiesSalesAuctionABI = [{"constant":false,"inputs":
[{"name":"_tokenId","type":"uint256"},
{"name":"_startingPrice","type":"uint256"},
{"name":"_endingPrice","type":"uint256"},
{"name":"_duration","type":"uint256"},
{"name":"_seller","type":"address"}], "name":"createAuction", "outp
uts":
[], "payable":false, "stateMutability":"nonpayable", "type":"functio
n"}, {"constant":false, "inputs": [], "name": "unpause", "outputs":
[{"name": "", "type": "bool"}], "payable": false, "stateMutability": "no
npayable", "type": "function"}, {"constant": false, "inputs":
[{"name": "_tokenId", "type": "uint256"}], "name": "bid", "outputs":
[], "payable": true, "stateMutability": "payable", "type": "function"},
{"constant": true, "inputs":
[{"name": "", "type": "uint256"}], "name": "lastGen0SalePrices", "outpu
ts":
[{"name": "", "type": "uint256"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": true, "inputs":
[], "name": "paused", "outputs":
[{"name": "", "type": "bool"}], "payable": false, "stateMutability": "vi
ew", "type": "function"}, {"constant": false, "inputs":
[], "name": "withdrawBalance", "outputs":
[], "payable": false, "stateMutability": "nonpayable", "type": "functio
n"}, {"constant": true, "inputs":
[{"name": "_tokenId", "type": "uint256"}], "name": "getAuction", "outpu
ts": [{"name": "seller", "type": "address"},
{"name": "startingPrice", "type": "uint256"},
{"name": "endingPrice", "type": "uint256"},
{"name": "duration", "type": "uint256"},
{"name": "startedAt", "type": "uint256"}], "payable": false, "stateMuta
bility": "view", "type": "function"}, {"constant": true, "inputs":
[], "name": "ownerCut", "outputs":
[{"name": "", "type": "uint256"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": false, "inputs":
[], "name": "pause", "outputs":
[{"name": "", "type": "bool"}], "payable": false, "stateMutability": "no
npayable", "type": "function"}, {"constant": true, "inputs":
[], "name": "isSaleClockAuction", "outputs":
[{"name": "", "type": "bool"}], "payable": false, "stateMutability": "vi
ew", "type": "function"}, {"constant": false, "inputs":
[{"name": "_tokenId", "type": "uint256"}], "name": "cancelAuctionWhenP
aused", "outputs":
[], "payable": false, "stateMutability": "nonpayable", "type": "functio
n"}, {"constant": true, "inputs":
[], "name": "gen0SaleCount", "outputs":
[{"name": "", "type": "uint256"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": true, "inputs":
[], "name": "owner", "outputs":
[{"name": "", "type": "address"}], "payable": false, "stateMutability":
```

```

"view", "type": "function"}, {"constant": false, "inputs":
[{"name": "_tokenId", "type": "uint256"}], "name": "cancelAuction", "ou
tputs":
[], "payable": false, "stateMutability": "nonpayable", "type": "functio
n"}, {"constant": true, "inputs":
[{"name": "_tokenId", "type": "uint256"}], "name": "getCurrentPrice", "
outputs":
[{"name": "", "type": "uint256"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": true, "inputs":
[], "name": "nonFungibleContract", "outputs":
[{"name": "", "type": "address"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": true, "inputs":
[], "name": "averageGen0SalePrice", "outputs":
[{"name": "", "type": "uint256"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": false, "inputs":
[{"name": "newOwner", "type": "address"}], "name": "transferOwnership"
, "outputs":
[], "payable": false, "stateMutability": "nonpayable", "type": "functio
n"}, {"inputs": [{"name": "_nftAddr", "type": "address"},
{"name": "_cut", "type": "uint256"}], "payable": false, "stateMutabilit
y": "nonpayable", "type": "constructor"},
{"anonymous": false, "inputs":
[{"indexed": false, "name": "tokenId", "type": "uint256"},
{"indexed": false, "name": "startingPrice", "type": "uint256"},
{"indexed": false, "name": "endingPrice", "type": "uint256"},
{"indexed": false, "name": "duration", "type": "uint256"}], "name": "Auc
tionCreated", "type": "event"}, {"anonymous": false, "inputs":
[{"indexed": false, "name": "tokenId", "type": "uint256"},
{"indexed": false, "name": "totalPrice", "type": "uint256"},
{"indexed": false, "name": "winner", "type": "address"}], "name": "Aucti
onSuccessful", "type": "event"}, {"anonymous": false, "inputs":
[{"indexed": false, "name": "tokenId", "type": "uint256"}], "name": "Auc
tionCancelled", "type": "event"}, {"anonymous": false, "inputs":
[], "name": "Pause", "type": "event"}, {"anonymous": false, "inputs":
[], "name": "Unpause", "type": "event"}];

```

// ABI 数据可以从智能合约链接中拷贝

```

var cryptoKittiesCoreABI = [{"constant": true, "inputs":
[{"name": "_interfaceID", "type": "bytes4"}], "name": "supportsInterfa
ce", "outputs":
[{"name": "", "type": "bool"}], "payable": false, "stateMutability": "vi
ew", "type": "function"}, {"constant": true, "inputs":
[], "name": "cfoAddress", "outputs":
[{"name": "", "type": "address"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": true, "inputs":
[{"name": "_tokenId", "type": "uint256"},
{"name": "_preferredTransport", "type": "string"}], "name": "tokenMeta
data", "outputs":
[{"name": "infoUrl", "type": "string"}], "payable": false, "stateMutabi
lity": "view", "type": "function"}, {"constant": true, "inputs":
[], "name": "promoCreatedCount", "outputs":
[{"name": "", "type": "uint256"}], "payable": false, "stateMutability":

```

```

"view", "type": "function"}, {"constant": true, "inputs":
[], "name": "name", "outputs":
[{"name": "", "type": "string"}], "payable": false, "stateMutability": "
view", "type": "function"}, {"constant": false, "inputs":
[{"name": "_to", "type": "address"},
{"name": "_tokenId", "type": "uint256"}], "name": "approve", "outputs":
[], "payable": false, "stateMutability": "nonpayable", "type": "functio
n"}, {"constant": true, "inputs": [], "name": "ceoAddress", "outputs":
[{"name": "", "type": "address"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": true, "inputs":
[], "name": "GEN0_STARTING_PRICE", "outputs":
[{"name": "", "type": "uint256"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": false, "inputs":
[{"name": "_address", "type": "address"}], "name": "setSiringAuctionAd
dress", "outputs":
[], "payable": false, "stateMutability": "nonpayable", "type": "functio
n"}, {"constant": true, "inputs": [], "name": "totalSupply", "outputs":
[{"name": "", "type": "uint256"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": true, "inputs":
[], "name": "pregnantKitties", "outputs":
[{"name": "", "type": "uint256"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": true, "inputs":
[{"name": "_kittyId", "type": "uint256"}], "name": "isPregnant", "outpu
ts":
[{"name": "", "type": "bool"}], "payable": false, "stateMutability": "vi
ew", "type": "function"}, {"constant": true, "inputs":
[], "name": "GEN0_AUCTION_DURATION", "outputs":
[{"name": "", "type": "uint256"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": true, "inputs":
[], "name": "siringAuction", "outputs":
[{"name": "", "type": "address"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": false, "inputs":
[{"name": "_from", "type": "address"},
{"name": "_to", "type": "address"},
{"name": "_tokenId", "type": "uint256"}], "name": "transferFrom", "outp
uts":
[], "payable": false, "stateMutability": "nonpayable", "type": "functio
n"}, {"constant": false, "inputs":
[{"name": "_address", "type": "address"}], "name": "setGeneScienceAddr
ess", "outputs":
[], "payable": false, "stateMutability": "nonpayable", "type": "functio
n"}, {"constant": false, "inputs":
[{"name": "_newCEO", "type": "address"}], "name": "setCEO", "outputs":
[], "payable": false, "stateMutability": "nonpayable", "type": "functio
n"}, {"constant": false, "inputs":
[{"name": "_newC00", "type": "address"}], "name": "setC00", "outputs":
[], "payable": false, "stateMutability": "nonpayable", "type": "functio
n"}, {"constant": false, "inputs":
[{"name": "_kittyId", "type": "uint256"},
{"name": "_startingPrice", "type": "uint256"},
{"name": "_endingPrice", "type": "uint256"},
{"name": "_duration", "type": "uint256"}], "name": "createSaleAuction"

```

```
, "outputs":
[[], "payable": false, "stateMutability": "nonpayable", "type": "function"}, {"constant": false, "inputs": [], "name": "unpause", "outputs":
[[], "payable": false, "stateMutability": "nonpayable", "type": "function"}, {"constant": true, "inputs":
[{"name": "", "type": "uint256"}], "name": "sireAllowedToAddress", "outputs":
[{"name": "", "type": "address"}], "payable": false, "stateMutability": "view", "type": "function"}, {"constant": true, "inputs":
[{"name": "_matronId", "type": "uint256"}, {"name": "_sireId", "type": "uint256"}], "name": "canBreedWith", "outputs":
[{"name": "", "type": "bool"}], "payable": false, "stateMutability": "view", "type": "function"}, {"constant": true, "inputs":
[{"name": "", "type": "uint256"}], "name": "kittyIndexToApproved", "outputs":
[{"name": "", "type": "address"}], "payable": false, "stateMutability": "view", "type": "function"}, {"constant": false, "inputs":
[{"name": "_kittyId", "type": "uint256"}, {"name": "_startingPrice", "type": "uint256"}, {"name": "_endingPrice", "type": "uint256"}, {"name": "_duration", "type": "uint256"}], "name": "createSiringAuction", "outputs":
[[], "payable": false, "stateMutability": "nonpayable", "type": "function"}, {"constant": false, "inputs":
[{"name": "val", "type": "uint256"}], "name": "setAutoBirthFee", "outputs":
[[], "payable": false, "stateMutability": "nonpayable", "type": "function"}, {"constant": false, "inputs":
[{"name": "_addr", "type": "address"}, {"name": "_sireId", "type": "uint256"}], "name": "approveSiring", "outputs":
[[], "payable": false, "stateMutability": "nonpayable", "type": "function"}, {"constant": false, "inputs":
[{"name": "_newCF0", "type": "address"}], "name": "setCF0", "outputs":
[[], "payable": false, "stateMutability": "nonpayable", "type": "function"}, {"constant": false, "inputs":
[{"name": "_genes", "type": "uint256"}, {"name": "_owner", "type": "address"}], "name": "createPromoKitty", "outputs":
[[], "payable": false, "stateMutability": "nonpayable", "type": "function"}, {"constant": false, "inputs":
[{"name": "secs", "type": "uint256"}], "name": "setSecondsPerBlock", "outputs":
[[], "payable": false, "stateMutability": "nonpayable", "type": "function"}, {"constant": true, "inputs": [], "name": "paused", "outputs":
[{"name": "", "type": "bool"}], "payable": false, "stateMutability": "view", "type": "function"}, {"constant": false, "inputs":
[[], "name": "withdrawBalance", "outputs":
[[], "payable": false, "stateMutability": "nonpayable", "type": "function"}, {"constant": true, "inputs":
[{"name": "_tokenId", "type": "uint256"}], "name": "ownerOf", "outputs":
```

```
:
[{"name":"owner","type":"address"}], "payable":false, "stateMutability":
"view", "type":"function"}, {"constant":true, "inputs":
[], "name":"GEN0_CREATION_LIMIT", "outputs":
[{"name":""," "type":"uint256"}], "payable":false, "stateMutability":
"view", "type":"function"}, {"constant":true, "inputs":
[], "name":"newContractAddress", "outputs":
[{"name":""," "type":"address"}], "payable":false, "stateMutability":
"view", "type":"function"}, {"constant":false, "inputs":
[{"name":"_address", "type":"address"}], "name":"setSaleAuctionAddr
ess", "outputs":
[], "payable":false, "stateMutability":"nonpayable", "type":"functio
n"}, {"constant":true, "inputs":
[{"name":"_owner", "type":"address"}], "name":"balanceOf", "outputs"
:
[{"name":"count", "type":"uint256"}], "payable":false, "stateMutabil
ity":"view", "type":"function"}, {"constant":false, "inputs":
[{"name":"_v2Address", "type":"address"}], "name":"setNewAddress", "
outputs":
[], "payable":false, "stateMutability":"nonpayable", "type":"functio
n"}, {"constant":true, "inputs":
[], "name":"secondsPerBlock", "outputs":
[{"name":""," "type":"uint256"}], "payable":false, "stateMutability":
"view", "type":"function"}, {"constant":false, "inputs":
[], "name":"pause", "outputs":
[], "payable":false, "stateMutability":"nonpayable", "type":"functio
n"}, {"constant":true, "inputs":
[{"name":"_owner", "type":"address"}], "name":"tokensOfOwner", "outp
uts":
[{"name":"ownerTokens", "type":"uint256[]"}], "payable":false, "stat
eMutability":"view", "type":"function"},
{"constant":false, "inputs":
[{"name":"_matronId", "type":"uint256"}], "name":"giveBirth", "outpu
ts":
[{"name":""," "type":"uint256"}], "payable":false, "stateMutability":
"nonpayable", "type":"function"}, {"constant":false, "inputs":
[], "name":"withdrawAuctionBalances", "outputs":
[], "payable":false, "stateMutability":"nonpayable", "type":"functio
n"}, {"constant":true, "inputs":[], "name":"symbol", "outputs":
[{"name":""," "type":"string"}], "payable":false, "stateMutability":
"view", "type":"function"}, {"constant":true, "inputs":
[{"name":""," "type":"uint256"}], "name":"cooldowns", "outputs":
[{"name":""," "type":"uint32"}], "payable":false, "stateMutability":
"view", "type":"function"}, {"constant":true, "inputs":
[{"name":""," "type":"uint256"}], "name":"kittyIndexToOwner", "output
s":
[{"name":""," "type":"address"}], "payable":false, "stateMutability":
"view", "type":"function"}, {"constant":false, "inputs":
[{"name":"_to", "type":"address"},
{"name":"_tokenId", "type":"uint256"}], "name":"transfer", "outputs"
:
[], "payable":false, "stateMutability":"nonpayable", "type":"functio
```



```

n"}, {"constant": true, "inputs": [], "name": "cooAddress", "outputs":
[{"name": "", "type": "address"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": true, "inputs":
[], "name": "autoBirthFee", "outputs":
[{"name": "", "type": "uint256"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": true, "inputs":
[], "name": "erc721Metadata", "outputs":
[{"name": "", "type": "address"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": false, "inputs":
[{"name": "_genes", "type": "uint256"}], "name": "createGen0Auction", "
outputs":
[], "payable": false, "stateMutability": "nonpayable", "type": "functio
n"}, {"constant": true, "inputs":
[{"name": "_kittyId", "type": "uint256"}], "name": "isReadyToBreed", "o
utputs":
[{"name": "", "type": "bool"}], "payable": false, "stateMutability": "vi
ew", "type": "function"}, {"constant": true, "inputs":
[], "name": "PROMO_CREATION_LIMIT", "outputs":
[{"name": "", "type": "uint256"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": false, "inputs":
[{"name": "_contractAddress", "type": "address"}], "name": "setMetadat
aAddress", "outputs":
[], "payable": false, "stateMutability": "nonpayable", "type": "functio
n"}, {"constant": true, "inputs": [], "name": "saleAuction", "outputs":
[{"name": "", "type": "address"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": true, "inputs":
[{"name": "_id", "type": "uint256"}], "name": "getKitty", "outputs":
[{"name": "isGestating", "type": "bool"},
{"name": "isReady", "type": "bool"},
{"name": "cooldownIndex", "type": "uint256"},
{"name": "nextActionAt", "type": "uint256"},
{"name": "siringWithId", "type": "uint256"},
{"name": "birthTime", "type": "uint256"},
{"name": "matronId", "type": "uint256"},
{"name": "sireId", "type": "uint256"},
{"name": "generation", "type": "uint256"},
{"name": "genes", "type": "uint256"}], "payable": false, "stateMutabili
ty": "view", "type": "function"}, {"constant": false, "inputs":
[{"name": "_sireId", "type": "uint256"},
{"name": "_matronId", "type": "uint256"}], "name": "bidOnSiringAuction
", "outputs":
[], "payable": true, "stateMutability": "payable", "type": "function"},
{"constant": true, "inputs": [], "name": "gen0CreatedCount", "outputs":
[{"name": "", "type": "uint256"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": true, "inputs":
[], "name": "geneScience", "outputs":
[{"name": "", "type": "address"}], "payable": false, "stateMutability":
"view", "type": "function"}, {"constant": false, "inputs":
[{"name": "_matronId", "type": "uint256"},
{"name": "_sireId", "type": "uint256"}], "name": "breedWithAuto", "outp
uts":
[], "payable": true, "stateMutability": "payable", "type": "function"},

```



```

{"inputs":
[], "payable": false, "stateMutability": "nonpayable", "type": "constructor"},
{"payable": true, "stateMutability": "payable", "type": "fallback"},
{"anonymous": false, "inputs":
[{"indexed": false, "name": "owner", "type": "address"},
{"indexed": false, "name": "matronId", "type": "uint256"},
{"indexed": false, "name": "sireId", "type": "uint256"},
{"indexed": false, "name": "cooldownEndBlock", "type": "uint256"}], "name": "Pregnant", "type": "event"}, {"anonymous": false, "inputs":
[{"indexed": false, "name": "from", "type": "address"},
{"indexed": false, "name": "to", "type": "address"},
{"indexed": false, "name": "tokenId", "type": "uint256"}], "name": "Transfer", "type": "event"}, {"anonymous": false, "inputs":
[{"indexed": false, "name": "owner", "type": "address"},
{"indexed": false, "name": "approved", "type": "address"},
{"indexed": false, "name": "tokenId", "type": "uint256"}], "name": "Approval", "type": "event"}, {"anonymous": false, "inputs":
[{"indexed": false, "name": "owner", "type": "address"},
{"indexed": false, "name": "kittyId", "type": "uint256"},
{"indexed": false, "name": "matronId", "type": "uint256"},
{"indexed": false, "name": "sireId", "type": "uint256"},
{"indexed": false, "name": "genes", "type": "uint256"}], "name": "Birth", "type": "event"}, {"anonymous": false, "inputs":
[{"indexed": false, "name": "newContract", "type": "address"}], "name": "ContractUpgrade", "type": "event"}];

var cryptoKittiesSalesAuctionAddress =
"0xb1690c08e213a35ed9bab7b318de14420fb57d8c";
var cryptoKittiesCoreAddress =
'0x06012c8cf97bead5deae237070f9587f8e7a266d';

// 根据合约地址构建合约
var salesAuction =
web3.eth.contract(cryptoKittiesSalesAuctionABI).at(cryptoKittiesSalesAuctionAddress);

var coreContract =
web3.eth.contract(cryptoKittiesCoreABI).at(cryptoKittiesCoreAddress);

// 笔者使用的 PHP 调用nodejs提供的一些http 接口，当然也可以全部使用nodejs写
var server = app.listen(8080, function ()
{
    var port = server.address().port;
    console.log('Example app listening at port:%s', port);
});

```

读取猫的代数 and 休息时间 cooldown_index

```

// 根据kitty_id对应猫的gen（代数）、休息时间索引（cooldown_index）
// 外部调用时，可以访问 http://127.0.0.1:8080/get_kitty/?
kitty_id=123
app.get('/get_kitty/', function (req, res)
{

    var kitty = coreContract.getKitty(req.param('kitty_id'));

    // 返回结果是一个数组，按参数在数组中的位置来取，并做适当类型转换
    var data = {errno: "0", gen:kitty[8].toString(10),
cooldown_index: kitty[2].toString(10)};

    res.json(data);
});

```

读取猫的拍卖信息

```

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

    // 根据猫的id获取拍卖信息
    var auction = salesAuction.getAuction(req.param('kitty_id'));

    // 一种判断猫是否在出售的临时方法，应该有更好的，因为笔者有时会遇到结果
    不一致的问题
    if(auction[0] == '0x')
    {
        res.json({errno: "0",is_onsale: 0});
    }
    else
    {
        // 将这些数据保存在本地数据后，就可以根据猫的价格公式，在本地计算猫
        在当前时间的价格了
        var data =
        {
            errno: "0",
            is_onsale: 1,
            starting_price: web3.fromWei(auction[1], 'ether'),
            ending_price: web3.fromWei(auction[2], 'ether'),
            duration: auction[3].toString(10),
            started_at: auction[4].toString(10),
            auction: auction
        };

        res.json(data);
    }

});

```

获取当前猫的总数量

// 如果使用批量生猫抢靓号猫，可以根据当前猫的总数来判断是否该批量生猫了，这样就不用值守在电脑前了

```
app.get('/total_supply/', function (req, res)
{
    var totalSupply = coreContract.totalSupply();
    var data = {errno: "0",total_supply: totalSupply};
    res.json(data);
});
```

通过程序买猫

所有涉及到往区块链写数据的操作，都需要消耗 ETH，需要有钱包支持。geth console 导入钱包的方法可以参考[这里](#)。

需要注意 geth 的选项，我使用的一个选项供参考：

```
--rpcapi admin,eth,web3,personal

// 传入猫 id、价格、交易 gas_price 来买猫
// 如果设置的价格比猫当时的实际价格高，系统会退回多余的 ETH
app.get('/bid/', function (req, res)
{
    var kitty_id = req.param('kitty_id');
    var price = req.param('price');
    var gas_price = req.param('gas_price', -1);

    console.log("kitty_id:" + kitty_id + ";price: " + price);

    web3.personal.unlockAccount('你的钱包地址','你的钱包密码');

    if(gas_price > 0)
    {
        salesAuction.bid(kitty_id,
            {
                from: '你的钱包地址',
                to: '0xb1690c08e213a35ed9bab7b318de14420fb57d8c',
                value: web3.toWei(price, "ether"),
                gas: 219827,
                gasPrice: web3.toWei(gas_price, "gwei")
            },

            function(err, result){
                if(err)
```

```

        {
            console.log(err);
            res.json({errno: "1",error: err});
        }
        else
        {
            console.log(result);
            res.json({errno: "0", tx_hash:result});
        }
    }
    );
}
// 使用系统默认gasPrice, 代码有点冗余, 可以优化下
else
{
    salesAuction.bid(kitty_id,
        {
            from: '你的钱包地址',
            to: '0xb1690c08e213a35ed9bab7b318de14420fb57d8c',
            value: web3.toWei(price, "ether"),
            gas: 219827
        },

        function(err, result){
            if(err)
            {
                console.log(err);
                res.json({errno: "1",error: err});
            }
            else
            {
                console.log(result);
                res.json({errno: "0", tx_hash:result});
            }
        }
    );
}
});

```

通过程序卖猫

```

app.get('/create_sale_auction/', function (req, res)
{
    var kitty_id = req.param('kitty_id');

    var starting_price_eth = req.param('starting_price_eth',1);
    var starting_price_wei = web3.toWei(starting_price_eth,
"ether");

    var ending_price_eth = req.param('ending_price_eth',1);

```

```

var ending_price_wei = web3.toWei(ending_price_eth, "ether");

var duration_days = req.param('duration_days',1);
var duration_sec = parseFloat(duration_days) * 86400;
duration_sec = parseInt(duration_sec);

var gas_price_gwei = req.param('gas_price', -1);
var gas_price_wei = 0;

if(gas_price_gwei < 0)
{
    gas_price_wei = web3.eth.gasPrice;
}
else
{
    gas_price_wei = web3.toWei(gas_price_gwei, "gwei");
}

var seller = '你的钱包地址';

web3.personal.unlockAccount(seller,"你的钱包密码");

coreContract.createSaleAuction(kitty_id,
starting_price_wei,ending_price_wei,duration_sec,
{
    from: seller,
    to: cryptoKittiesCoreAddress,
    value: 0,
    gas: 219827,
    gasPrice: gas_price_wei
},

function(err, result){
    if(err)
    {
        console.log(err);
        res.json({errno: "1",error: err});
    }
    else
    {
        console.log(result);
        res.json({errno: "0", tx_hash:result});
    }
}

);
});

```

通过程序生猫

```

app.get('/breed_with_auto/', function (req, res)
{
    var matron_id = req.param('matron_id');
    var sire_id = req.param('sire_id');

    var gas_price_gwei = req.param('gas_price', -1);
    var gas_price_wei = 0;

    if(gas_price_gwei < 0)
    {
        gas_price_wei = web3.eth.gasPrice;
    }
    else
    {
        gas_price_wei = web3.toWei(gas_price_gwei, "gwei");
    }

    web3.personal.unlockAccount('你的钱包地址',"你的钱包密码");

    coreContract.breedWithAuto(matron_id, sire_id,
    {
        from: '你的钱包地址',
        to: '0x06012c8cf97bead5deae237070f9587f8e7a266d',
        // 目前的生猫费用是0.008 ETH, 会奖励给猫的接生婆
        value: web3.toWei(0.008, "ether"),
        gas: 219827,
        gasPrice: gas_price_wei
    },

    function(err, result){
        if(err)
        {
            console.log(err);
            res.json({errno: "1",error: err});
        }
        else
        {
            console.log(result);
            res.json({errno: "0", tx_hash:result});
        }
    }
    );
});

```

检查新挂单的猫咪

为了第一时间获取某只猫开卖了，可以通过监听 AuctionCreated 事件来获取开卖信息。此外，还可以监听猫的售出事件，拍卖取消事件。


```

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

    // 指定某个块高度，方便外部逐步遍历
    var blockNumber = req.param('block_number',-1);

    if(blockNumber < 0 )
    {
        res.json({errno: "1", error: 'wrong block_number'});
        return ;
    }

    var events = salesAuction.AuctionCreated({}, {fromBlock:
blockNumber, toBlock: blockNumber});

    var timestamp = web3.eth.getBlock(blockNumber).timestamp;

    events.get(function(error, result)
    {
        if(!error)
        {
            var count = result.length;
            var arr = [];
            for(var i = 0; i < count; i++)
            {
                var r = result[i];

                // 拍卖相关信息
                var transactionHash = r.transactionHash;
                var kittyId = r.args.tokenId.toString();
                var duration = r.args.duration;
                var startingPrice =
web3.fromWei(r.args.startingPrice, 'ether').toString();
                var endingPrice =
web3.fromWei(r.args.endingPrice, 'ether').toString();

                arr[i] =
                {
                    transaction_hash: transactionHash,
                    starting_price: startingPrice,
                    ending_price: endingPrice,
                    timestamp: timestamp,
                    duration: duration,
                    kitty_id: kittyId
                };
            }
        }
    })
}

```

```
        var data = {errno: "0", count: count, arr: arr };

        res.json(data);
        //console.log(data);
    }
    else
    {
        console.log(error);
    }
});

events.stopWatching();

});
```

使用 api.cryptokitties.co 读取数据

猫的部分数据没保存在区块链上，需要使用官方提供的 API 读取。下面举两个例子：

1. 按一定排序规则读取在售的猫咪，访问[这里](#)获取 API。
2. 读取某只猫咪的数据，请使用[该 API](#)。

通过程序赚 ETH

可以通过程序低买高卖、批量生猫抢靓号、当猫的接生婆赚 ETH，友情提示投机有风险，程序也可能有 Bug，投资需谨慎。

低买高卖

有了上面的基础，通过程序可以统计到所有历史成交记录，也可以观察到最新的成交记录。有一定的观察后，你会掌握一些猫的市场价格，比如目前（2018-04-03）编号在 3000 以内的猫的价格基本都在 1 ETH 以上，如果有人新挂出一只编号在 3000 以内且价格仅 0.1 的猫，那么你应该立即买入，买入后再以一个符合市场的常规价格挂单卖出。

批量生猫抢靓号

每新生一个猫，猫 ID 会在当前猫的最大 ID 基础上加 1 得到新猫的 ID。有一些玩家会通过批量生猫的方式提高抢到靓号猫的概率。如在当前猫的最大 ID 为 499900 的时候，使用 100 只恢复时间短的猫做猫妈妈，另外 100 只猫做为对应的猫妈妈，快速生出 100 只猫，会有较大的概率抢到编号为 500000 的猫。根据我的观察，编号为 500000 的主人的确是通过批量生猫抢到这个编号的。

选择合适的基因生出高价的猫

生猫是项基因工程，如果选择合适的猫作为爸妈，可以花少量的钱生出价值高的猫。关于基因工程，可以参考下面两篇文章：

- [《变异》](#)
- [《生育结果》](#)

当猫的接生婆赚 ETH

迷恋猫为了将游戏设计的更加逼真，猫怀孕后，需要一定时间后才能生猫。而以太坊智能合约是不能设置特定时间调用某个函数的，只能通过外部调用来触发生猫函数 `giveBirth`。官方为了鼓励第三方参与，将生猫函数设为公开的了，谁都可以调用，如果谁成功调用了，就可以获得接生费用，目前是 0.008 ETH。

下面是智能合约中的接生函数：

```
/// @notice Have a pregnant Kitty give birth!
/// @param _matronId A Kitty ready to give birth.
/// @return The Kitty ID of the new kitten.
/// @dev Looks at a given Kitty and, if pregnant and if the
gestation period has passed,
/// combines the genes of the two parents to create a new
kitten. The new Kitty is assigned
/// to the current owner of the matron. Upon successful
completion, both the matron and the
/// new kitten will be ready to breed again. Note that
anyone can call this function (if they
/// are willing to pay the gas!), but the new kitten always
goes to the mother's owner.
function giveBirth(uint256 _matronId)
external
whenNotPaused
returns(uint256)
{
    // Grab a reference to the matron in storage.
    Kitty storage matron = kitties[_matronId];

    // Check that the matron is a valid cat.
    require(matron.birthTime != 0);

    // Check that the matron is pregnant, and that its time
has come!
    require(_isReadyToGiveBirth(matron));

    // Grab a reference to the sire in storage.
    uint256 sireId = matron.siringWithId;
```

```

        Kitty storage sire = kitties[sireId];

        // Determine the higher generation number of the two
parents
        uint16 parentGen = matron.generation;
        if (sire.generation > matron.generation) {
            parentGen = sire.generation;
        }

        // Call the sooper-sekret gene mixing operation.
        uint256 childGenes = geneScience.mixGenes(matron.genes,
sire.genes, matron.cooldownEndBlock - 1);

        // Make the new kitten!
        address owner = kittyIndexToOwner[_matronId];
        uint256 kittenId = _createKitty(_matronId,
matron.siringWithId, parentGen + 1, childGenes, owner);

        // Clear the reference to sire from the matron (REQUIRED!
Having siringWithId
        // set is what marks a matron as being pregnant.)
        delete matron.siringWithId;

        // Every time a kitty gives birth counter is decremented.
pregnantKitties--;

        // Send the balance fee to the person who made birth
happen.
        msg.sender.send(autoBirthFee);

        // return the new kitten's ID
        return kittenId;
    }

```

一个怀孕的猫符合生猫条件后，只能被一个外部调用成功触发生猫。这意味着如果你失败了，就白白浪费了以太坊网络手续费。giveBirth 函数一次只能触发一个猫，为了提高效率，你可以编写一个智能合约在链上批量调用 giveBirth。另外，还会涉及到检查符合条件的猫，建议在非常熟悉游戏规则后再尝试通过生猫赚钱。

迷恋猫的一些统计数据

我通过读取智能合约数据，做了一些数据统计，截止到2018年03月10白的数据如下：

- 18 号猫以 253 ETH（约¥109万）成交，有 32 只猫的售价超过100 ETH；
- 约 6 万人买过猫，根据玩家钱包地址去重统计；
- 平台共有 61 万只猫，其中约1万只是平台发布的 0 代猫；
- 总交易额 3.96万 ETH（约¥1.7亿），总交易次数 29.8 万；

- 官方卖 0 代猫收入 = 8611 ETH;
- 官方手续费收入 = 712 ETH (手续费为交易额的3.75%) ;
- 官方总收入 = 卖 0 代猫收入 + 手续费收入 = 9324 ETH。

玩谜恋猫的一些启发

- 用户钱包即 ID, 开发者可以利用良好的以太坊生态开发 DApp;
- 一些用户因为想玩这个游戏, 学会了钱包的使用, 到 etherscan.io 查看交易数据, 让用户在玩的过程中也了解了区块链;
- 猫的交易达成时, 通过智能合约自动把钱转给卖家, 这和我们传统的交易方式完全不同, 不用担心服务提供方恶意侵占钱款;
- 禀赋效应: 是指当个人一旦拥有某项物品, 那么他对该物品价值的评价要比未拥有之前大大增加。但在实际过程中, 把猫的价格挂的太高了, 往往很难卖掉, 建议根据猫的稀有程度和历史成交数据客观评估价格, 让自己的猫流通起来;
- 社区很重要, 玩家通过在 Discord、Reddit、QQ群、微信群等地方交流后, 更容易对一些猫的价格达成共识, 还可以交流基因工程、生猫经验等有趣的地方;
- 目前的基础工具还不够完善, 玩家门槛有点高, 如使用 Chrome 浏览器玩时, 需要先想办法安装 MetaMask 钱包插件。基础工具(如钱包)的建设也是机会, 好基础工具可以获取到区块链用户流量资源。

开发时遇到问题怎么办?

如果在开发中遇到问题, 请先 Google 一下, 如果多次尝试后还未能解决, 可以加我微信 (微信号: yuange1024), 加好友时请注明区块链。