创建和发布以太坊代币

财经作家吴晓波说：“如果2018年有人跟你谈无人驾驶、人工智能、区块链、人类永生，记住，他们一半以上都是骗子，大规模的泡沫和骗子存在的地方。”

相关声明和资讯

看到标题，会不会有人想到通过发布自己的代币，然后再去某宝花几百几千块钱找个区块链白皮书代写，来进行ICO融资。然后"相信用不了多久，就会升职加薪当上总经理，出任CEO，迎娶白富美，走向人生巅峰。"
换作以前，可能还真有人这么做过吧。

先来看看相关资讯。
2017年9月4日下午，央行等七部委联合发布公告：ICO是未经批准非法融资行为。
2018年1月2日，人人公司发布RRCoin白皮书，要为社交网络打造一个开源的区块链平台——人人坊，并成立了RRCoin基金会。随之，股价两日暴涨67.02%。1月8日，据链科技消息称，监管部门前一天晚上已经约谈了人人网，RRcoin项目已经确定翻车。
2018年3月，某宝已全面下架了"区块链白皮书代写"相关商品和店铺。虚拟代币ICO乱象丛生，许多ICO都是旁氏骗局。

因此，本文从技术角度来介绍如何创建和发布基于以太坊的代币，从而揭开代币的神秘面纱。

ERC20代币介绍

在区块链中，代币指的是加密数字货币，不依靠法定货币机构发行，不受央行管控。

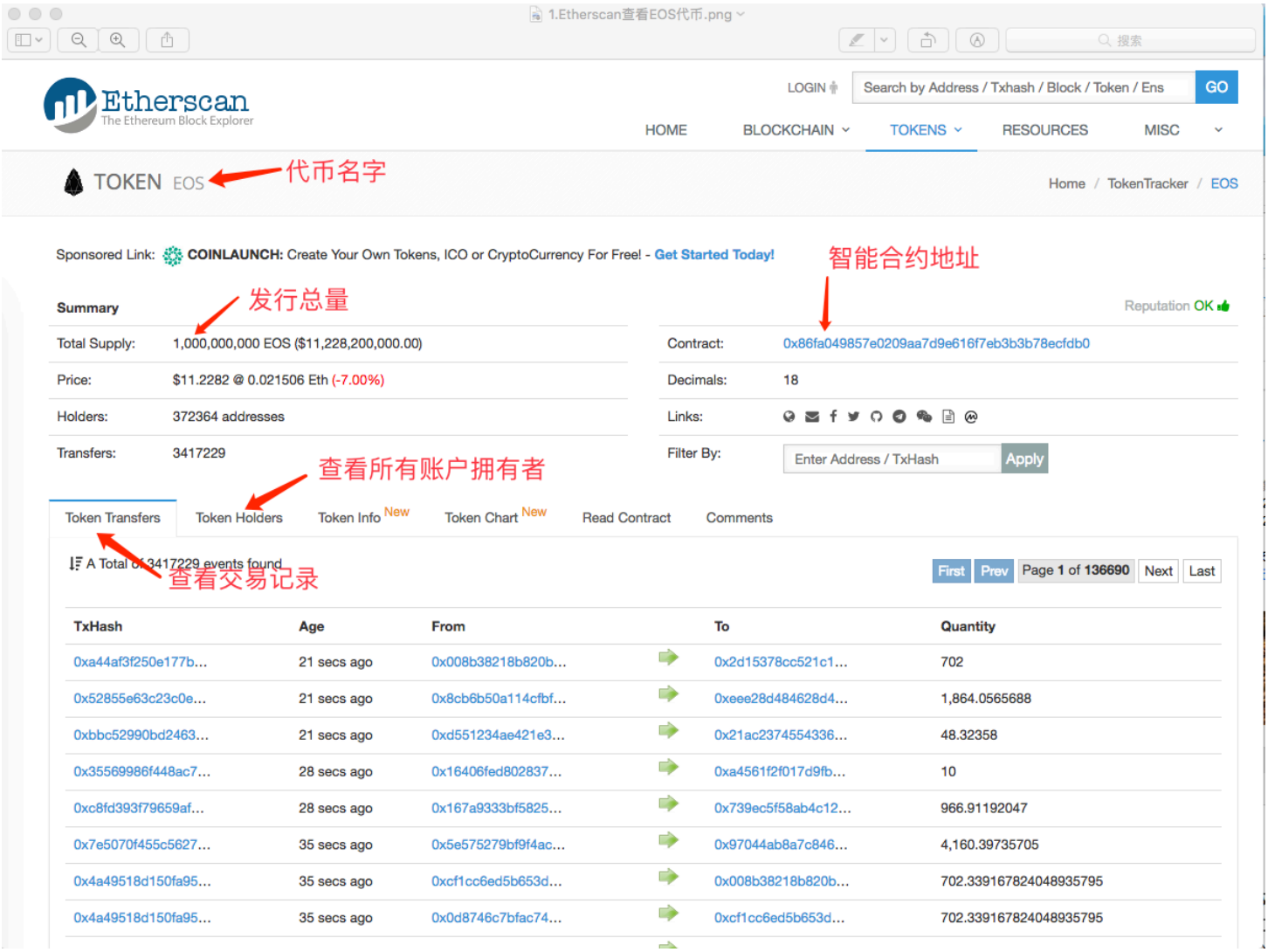
代币主要分为两种类型：
（1）有自己区块链项目型的代币：比特币、莱特币、瑞波币、以太币等，主要用于矿工奖励和防止垃圾交易

（2）基于以太坊去中心化智能合约平台的平台型代币，市面上绝大部分属于该分类：EOS、QTUM、OmiseGo等

通过以太坊平台，可以快速地创建遵循ERC20协议（有新的ERC23协议）的代币。

Etherscan 是 2015 年推出的一个以太坊区块探索和分析的分布式智能合同平台，用户可以使用其查看自己的交易详情以及以太坊中的任何信息，类似于"快递查询"的应用。
网址： https://etherscan.io/

如图，查看EOS代币的信息：



ERC20接口协议

ERC20是以太坊定义的一个代币标准，定义了实现代币时必须要遵守的协议，如指名代币名称、发行代币总量、查看对应账号的代币余额、代币交易等，实现对应的函数，如下：

```
// Abstract contract for the full ERC 20 Token standard
// https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
pragma solidity ^0.4.21;

contract EIP20Interface {
    /* This is a slight change to the ERC20 base standard.
    function totalSupply() constant returns (uint256 supply);
    is replaced with:
    uint256 public totalSupply;
    This automatically creates a getter function for the totalSupply.
    This is moved to the base contract since public getter functions are not
    currently recognised as an implementation of the matching abstract
    function by the compiler.
    */
    /// total amount of tokens
```

```
    uint256 public totalSupply;

    /// @param _owner The address from which the balance will be retrieved
    /// @return The balance
    function balanceOf(address _owner) public view returns (uint256 balance)
;

    /// @notice send `_value` token to `_to` from `msg.sender`
    /// @param _to The address of the recipient
    /// @param _value The amount of token to be transferred
    /// @return Whether the transfer was successful or not
    function transfer(address _to, uint256 _value) public returns (bool succ
ess);

    /// @notice send `_value` token to `_to` from `_from` on the condition i
t is approved by `_from`
    /// @param _from The address of the sender
    /// @param _to The address of the recipient
    /// @param _value The amount of token to be transferred
    /// @return Whether the transfer was successful or not
    function transferFrom(address _from, address _to, uint256 _value) public
 returns (bool success);

    /// @notice `msg.sender` approves `_spender` to spend `_value` tokens
    /// @param _spender The address of the account able to transfer the toke
ns
    /// @param _value The amount of tokens to be approved for transfer
    /// @return Whether the approval was successful or not
    function approve(address _spender, uint256 _value) public returns (bool
success);

    /// @param _owner The address of the account owning tokens
    /// @param _spender The address of the account able to transfer the toke
ns
    /// @return Amount of remaining tokens allowed to spent
    function allowance(address _owner, address _spender) public view returns
 (uint256 remaining);

    // solhint-disable-next-line no-simple-event-func-name
    event Transfer(address indexed _from, address indexed _to, uint256 _valu
e);
    event Approval(address indexed _owner, address indexed _spender, uint256
 _value);
}
```

注：官方已经把原来的ERC名字都改为了EIP。

Github地址：

```
https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
```

ERC20不是完美的，有待完善。美链BEC曾出现过ERC20漏洞事件，损失千亿。 ERC223是在ERC20的标准上引入了一个新功能，以防止意外转移的发生。

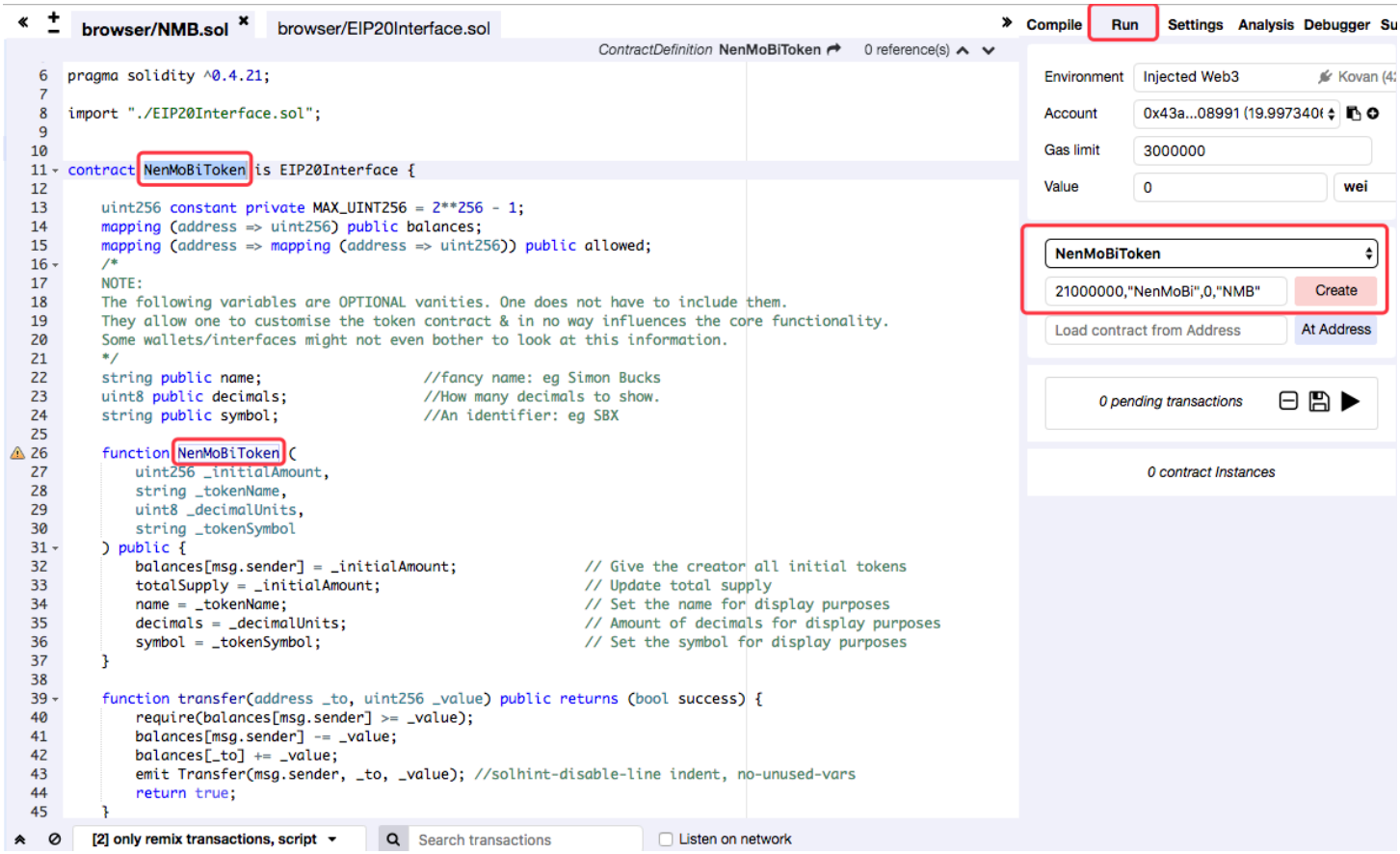
编写ERC20代币智能合约

网上有很多实现了ERC20接口的Demo，以下是Github上面的。

```
/*
Implements EIP20 token standard: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
.*/


pragma solidity ^0.4.21;

import "./EIP20Interface.sol";


contract EIP20 is EIP20Interface {

    uint256 constant private MAX_UINT256 = 2**256 - 1;
    mapping (address => uint256) public balances;
    mapping (address => mapping (address => uint256)) public allowed;
    /*
    NOTE:
    The following variables are OPTIONAL vanities. One does not have to include them.
    They allow one to customise the token contract & in no way influences the core functionality.
    Some wallets/interfaces might not even bother to look at this information.
    */
    string public name;                   //fancy name: eg Simon Bucks
    uint8 public decimals;                //How many decimals to show.
    string public symbol;                 //An identifier: eg SBX

    function EIP20(
        uint256 _initialAmount,
        string _tokenName,
        uint8 _decimalUnits,
        string _tokenSymbol
    ) public {
        balances[msg.sender] = _initialAmount;               // Give the cre
```

```
ator all initial tokens
        totalSupply = _initialAmount;                       // Update total
 supply
        name = _tokenName;                                  // Set the name
 for display purposes
        decimals = _decimalUnits;                           // Amount of de
cimals for display purposes
        symbol = _tokenSymbol;                              // Set the symb
ol for display purposes
    }

    function transfer(address _to, uint256 _value) public returns (bool succ
ess) {
        require(balances[msg.sender] >= _value);
        balances[msg.sender] -= _value;
        balances[_to] += _value;
        emit Transfer(msg.sender, _to, _value); //solhint-disable-line inden
t, no-unused-vars
        return true;
    }

    function transferFrom(address _from, address _to, uint256 _value) public
 returns (bool success) {
        uint256 allowance = allowed[_from][msg.sender];
        require(balances[_from] >= _value && allowance >= _value);
        balances[_to] += _value;
        balances[_from] -= _value;
        if (allowance < MAX_UINT256) {
            allowed[_from][msg.sender] -= _value;
        }
        emit Transfer(_from, _to, _value); //solhint-disable-line indent, no
-unused-vars
        return true;
    }

    function balanceOf(address _owner) public view returns (uint256 balance)
 {
        return balances[_owner];
    }

    function approve(address _spender, uint256 _value) public returns (bool
success) {
        allowed[msg.sender][_spender] = _value;
        emit Approval(msg.sender, _spender, _value); //solhint-disable-line
indent, no-unused-vars
        return true;
    }
```

```
    function allowance(address _owner, address _spender) public view returns
  (uint256 remaining) {
        return allowed[_owner][_spender];
    }
}
```

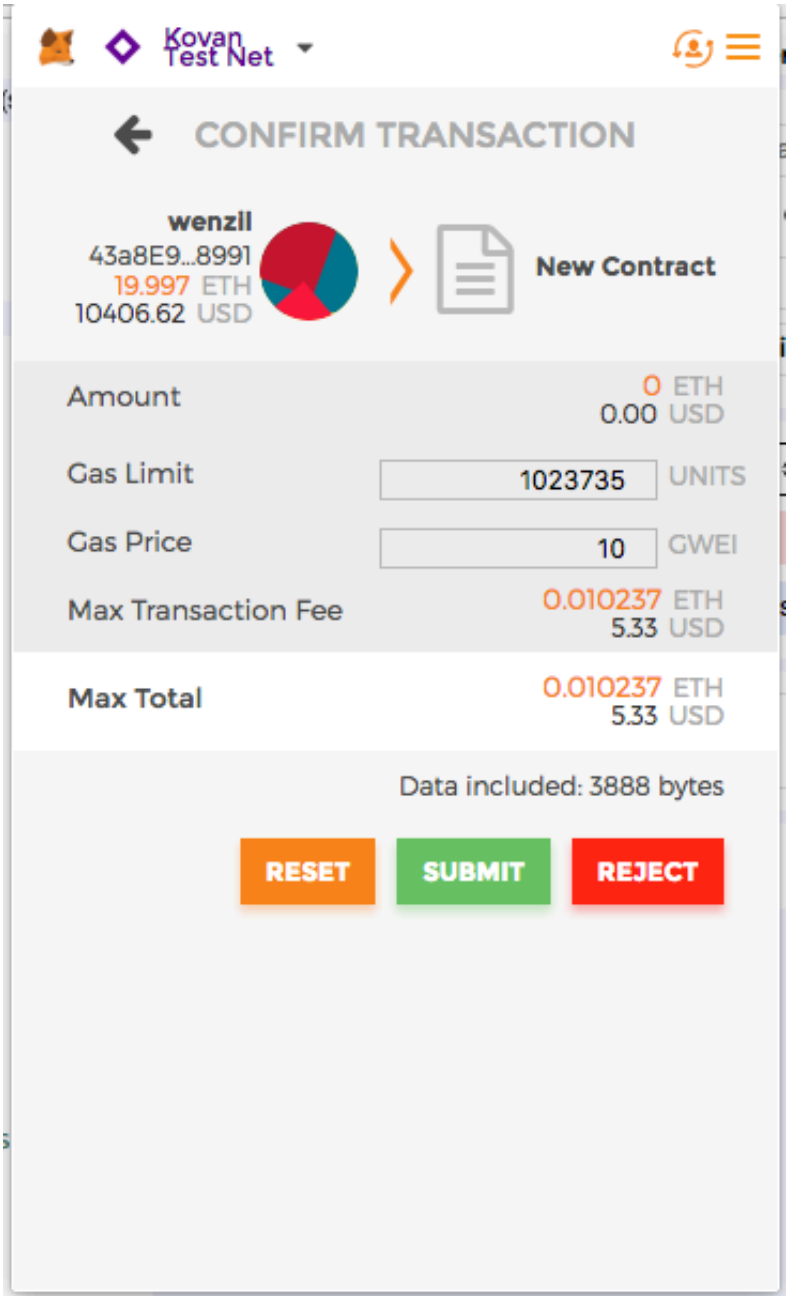


打开Remix Solidity IDE，复制上面的EIP20Interface和EIP20，将EIP20.sol的合约名称和构造函数修改为NenMoBiToken（嫩模币）。"Create"输入框输入：21000000,"NenMoBi",0,"NMB"，发布2100万个NMB（嫩模币），然后点击"Create"创建合约。

为什么取这个名字？因为之前看到一个兄弟在简书文章慷慨激昂，还有"梭哈老头"的那句话——“赢了会所嫩模,输了下海干活!”。

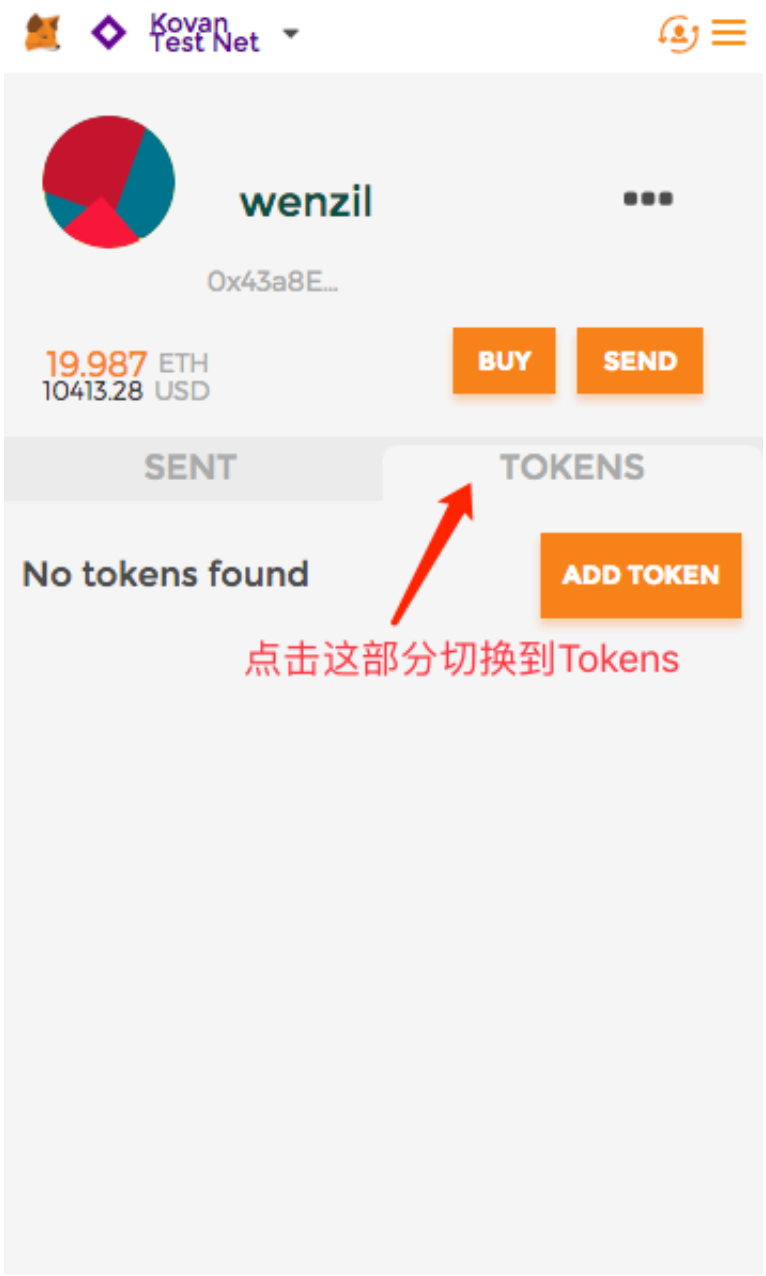
那位兄弟的简书文章《ICO（虚拟币众筹）到底是个什么东西》，很有意思，值得一看，里面提到了NMB（嫩模币）。所以我来探索下这位兄弟说的发布代币，揭开代币的神秘面纱。

话题有点扯远了，点击"Create"按钮之后会弹出MetaMask提交交易，点击"SUBMIT"进行提交。

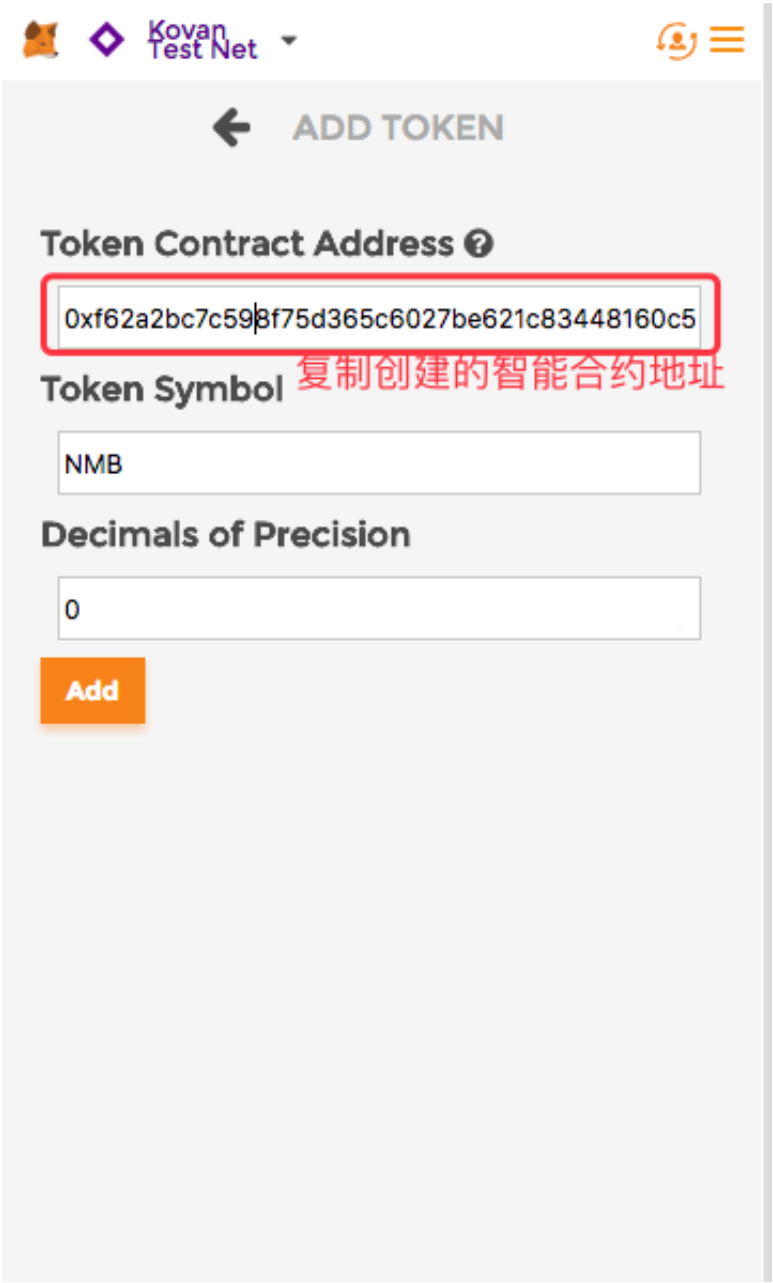


查看ERC20代币

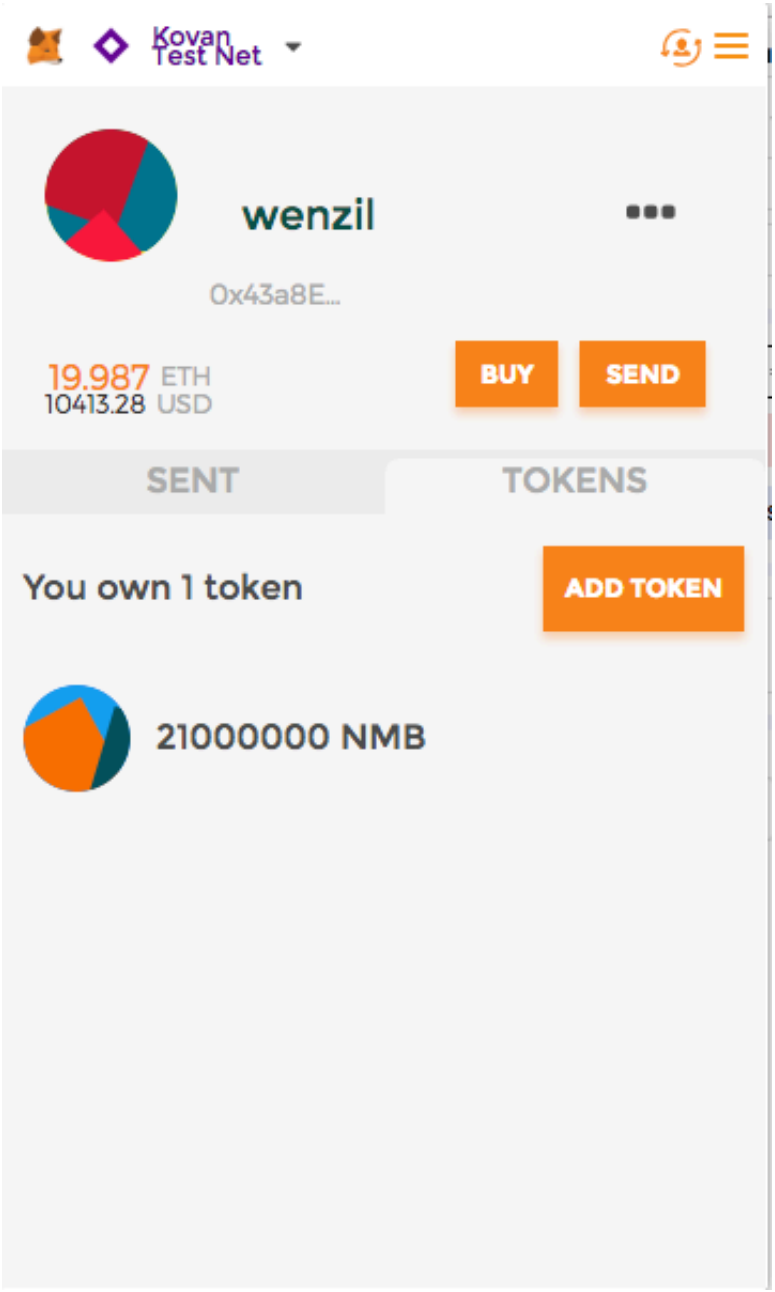
提交交易成功之后，打开MetaMask，切换到"Tokens"菜单。



点击"Add Token"按钮，跳转到如下页面，复制合约地址到第一个输入框，另外两个输入框是自动填写的，如图：



点击"Add"按钮，这时再查看"Tokens"发现该账号拥有了2100万个NMB（嫩模币）。
于是，可以通过Etherscan来查看刚刚部署的代币。



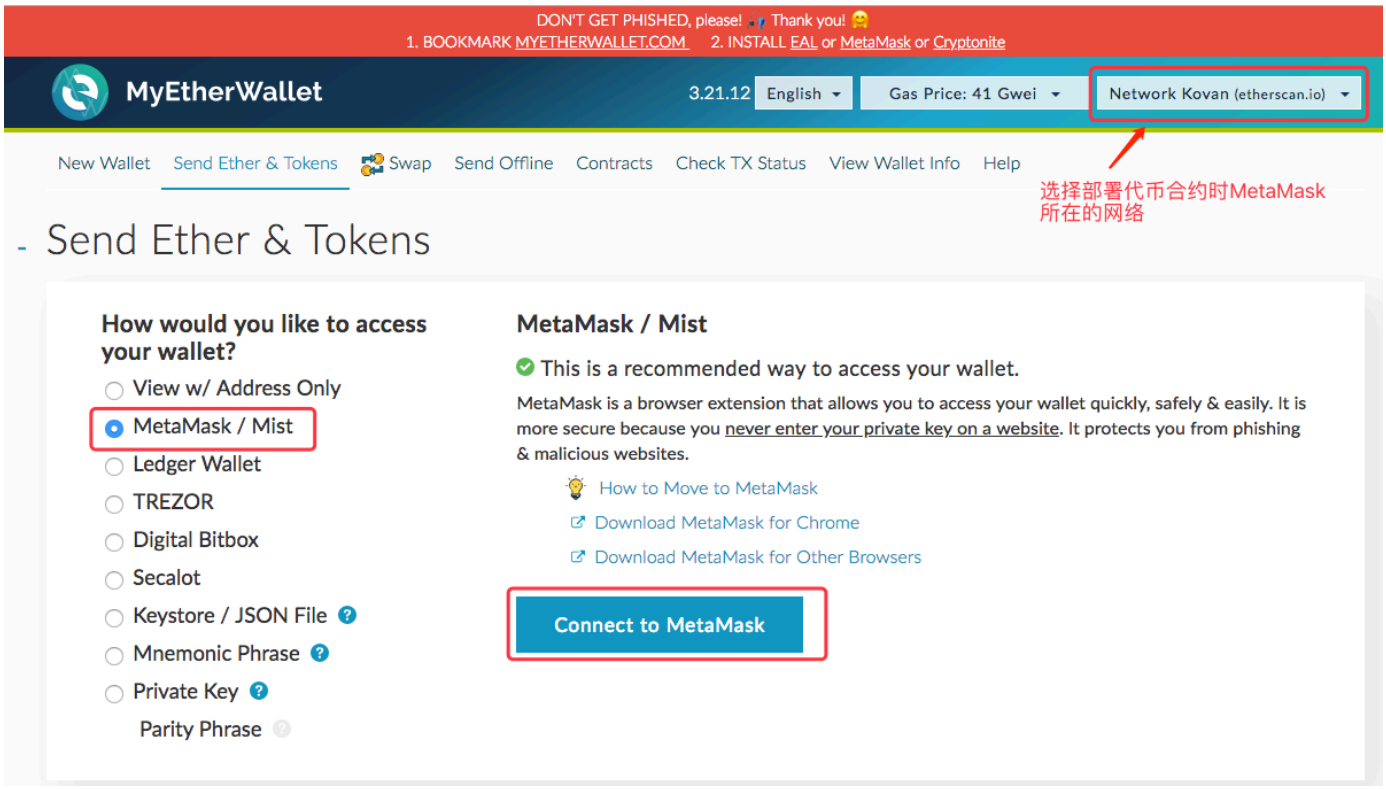
再次声明：这里创建的代币都是没有实际应用的空气币，只是用于学习和娱乐。也不鼓励发行无意义的空气币，别想着用来ICO，国家已禁止ICO。

代币交易

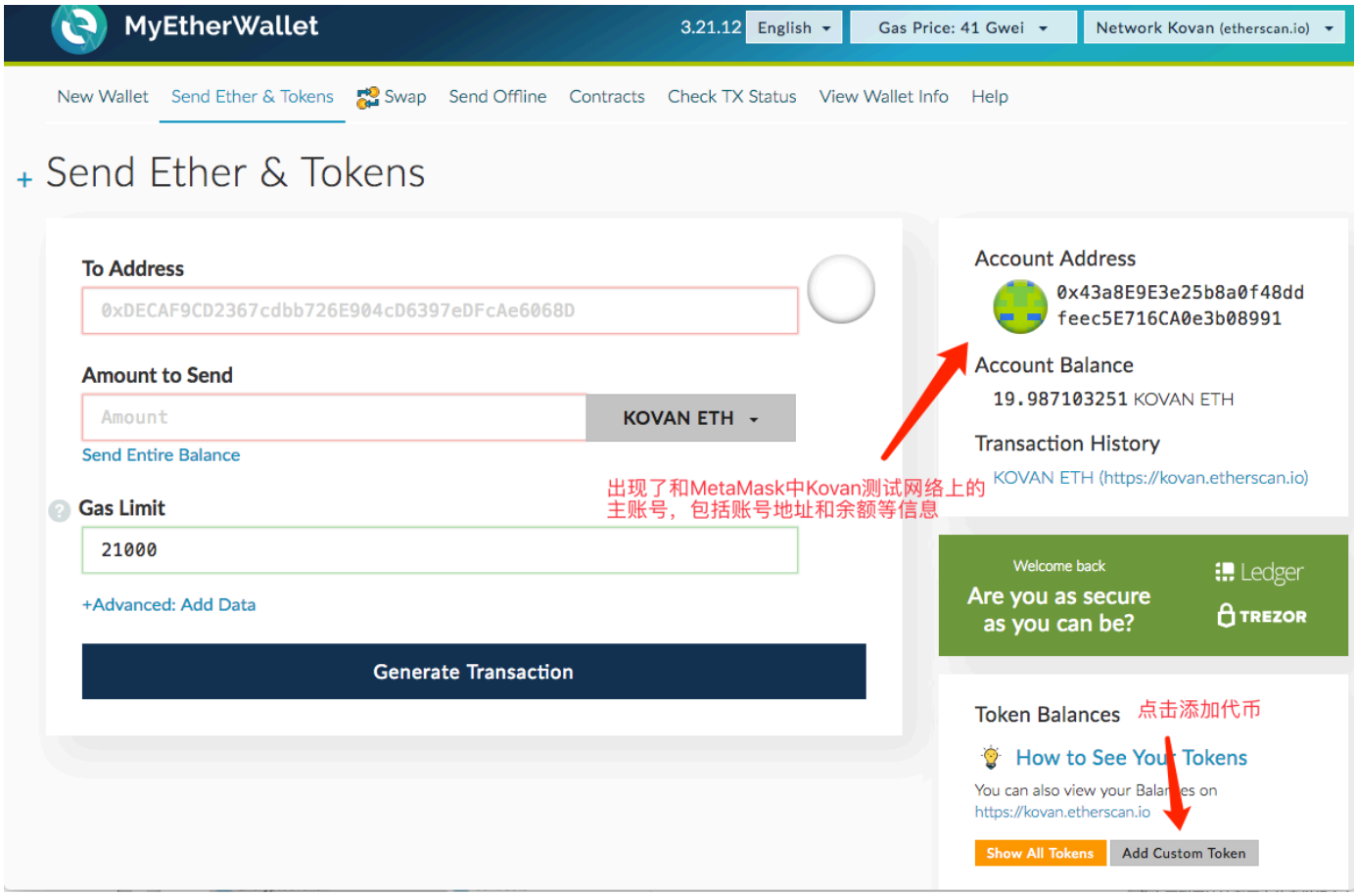
- MyEtherWallet和MetaMask结合使用

  MetaMask插件不能进行代币的交易，可以通过MyEtherWallet来交易。

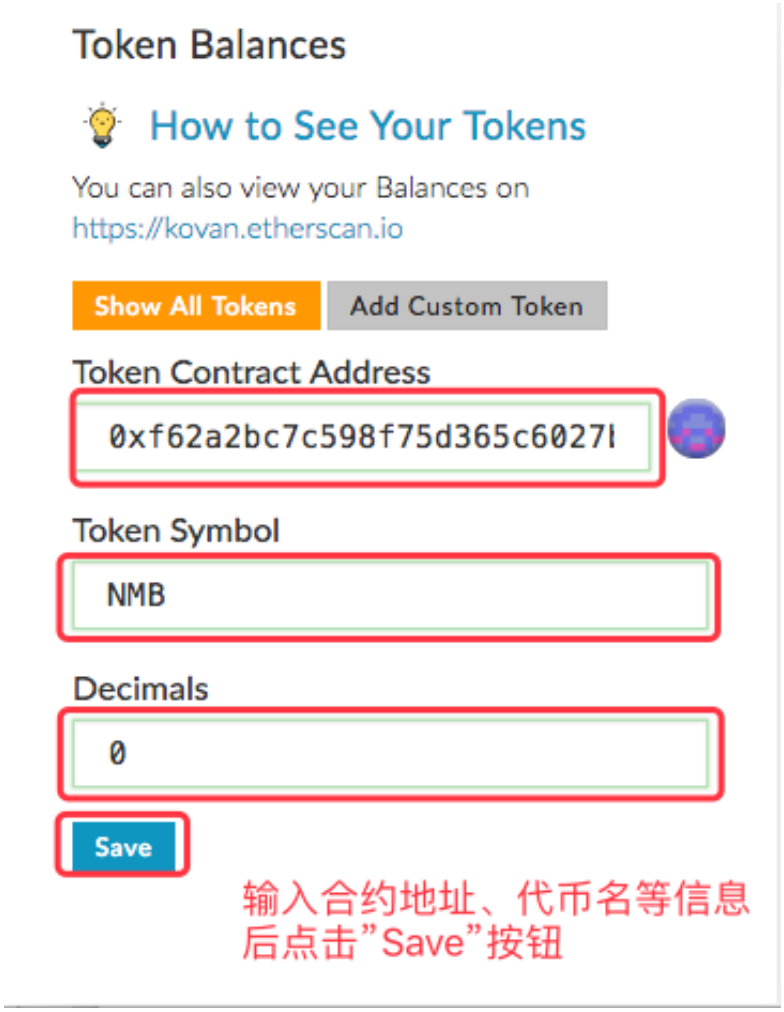
  进入页面后，按照下图来设置:



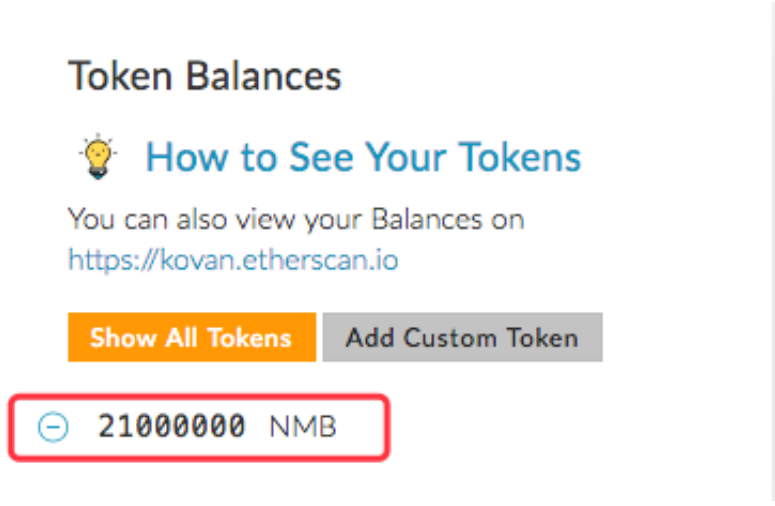
点击连接按钮，会跳转到如下页面：



点击添加代币按钮后，会弹出如下输入框：



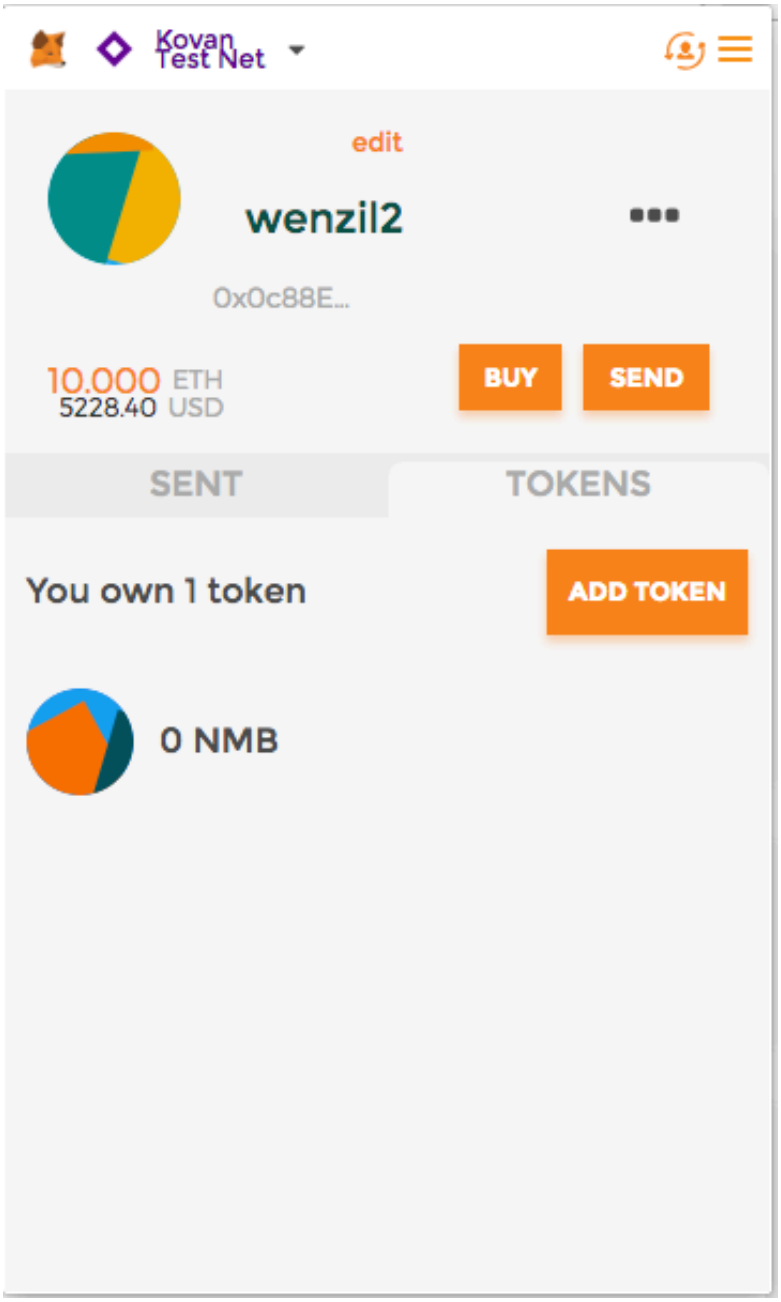
点击"Save"按钮之后，显示如下：



- 转入代币

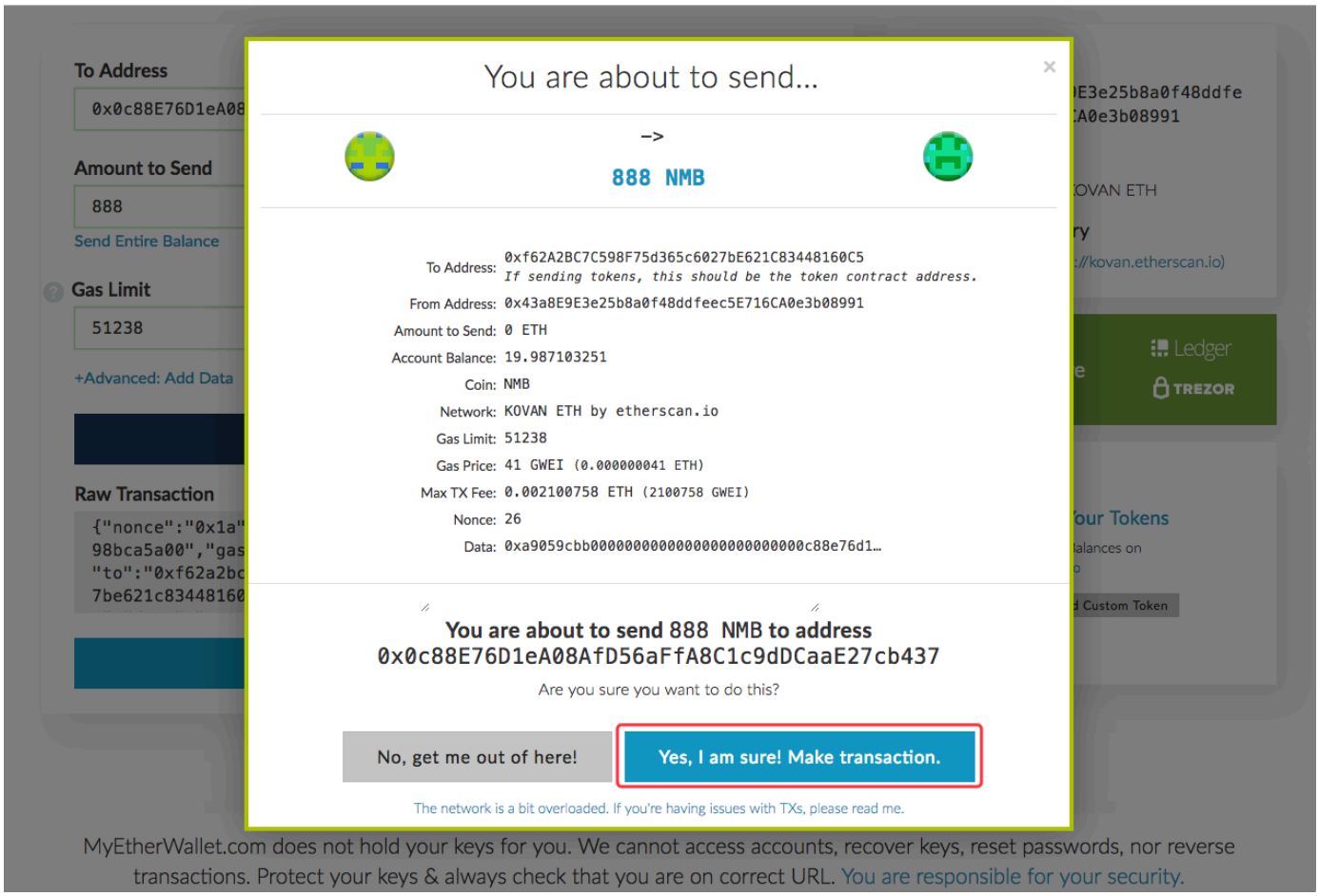
  以下演示向某个账户转入一定量的代币

  Kovan测试网络还有另外一个账号，如图:

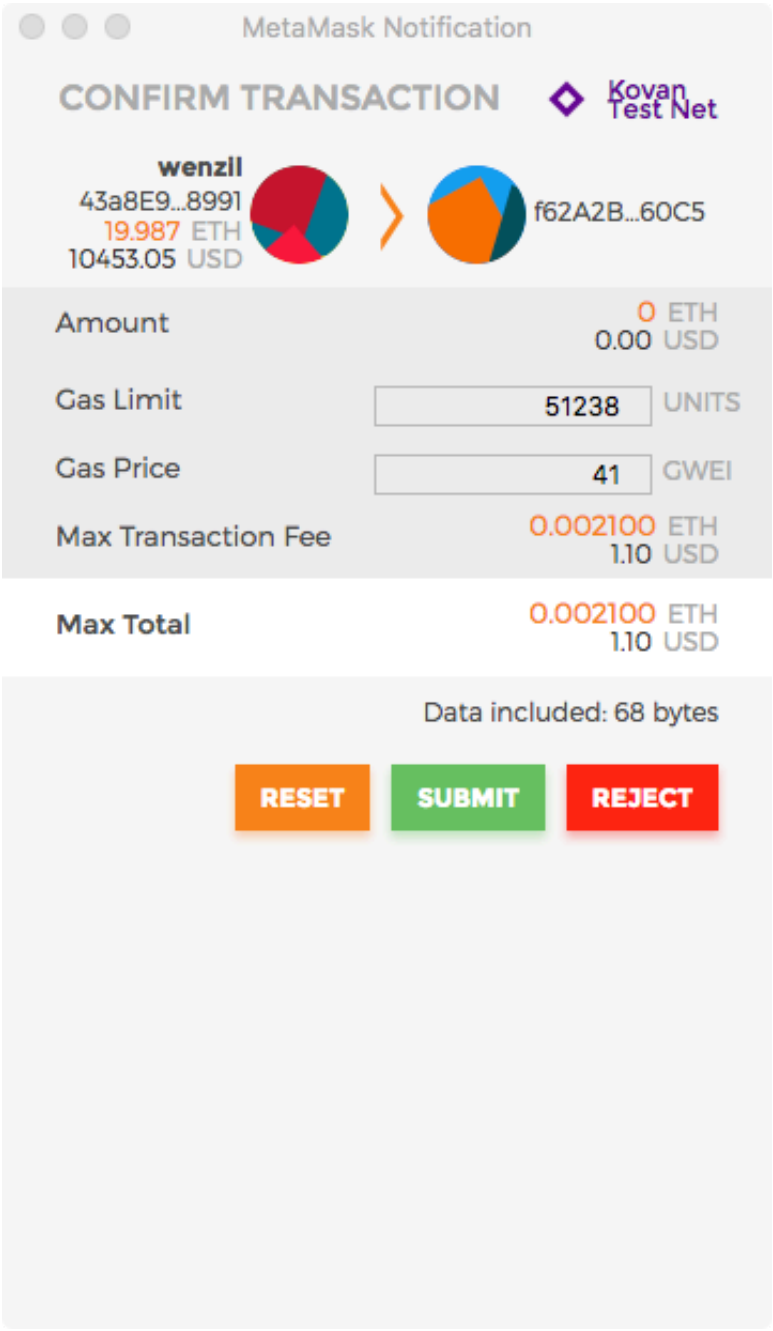


复制该账号的地址，然后在"To Address"输入框粘贴。

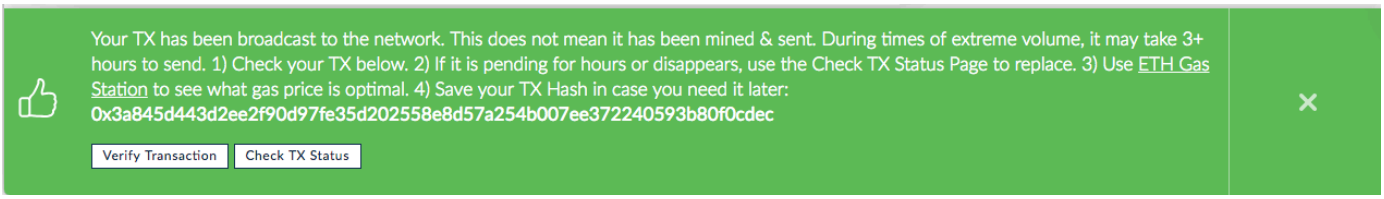
点击发送交易按钮，会弹出如下图所示内容:



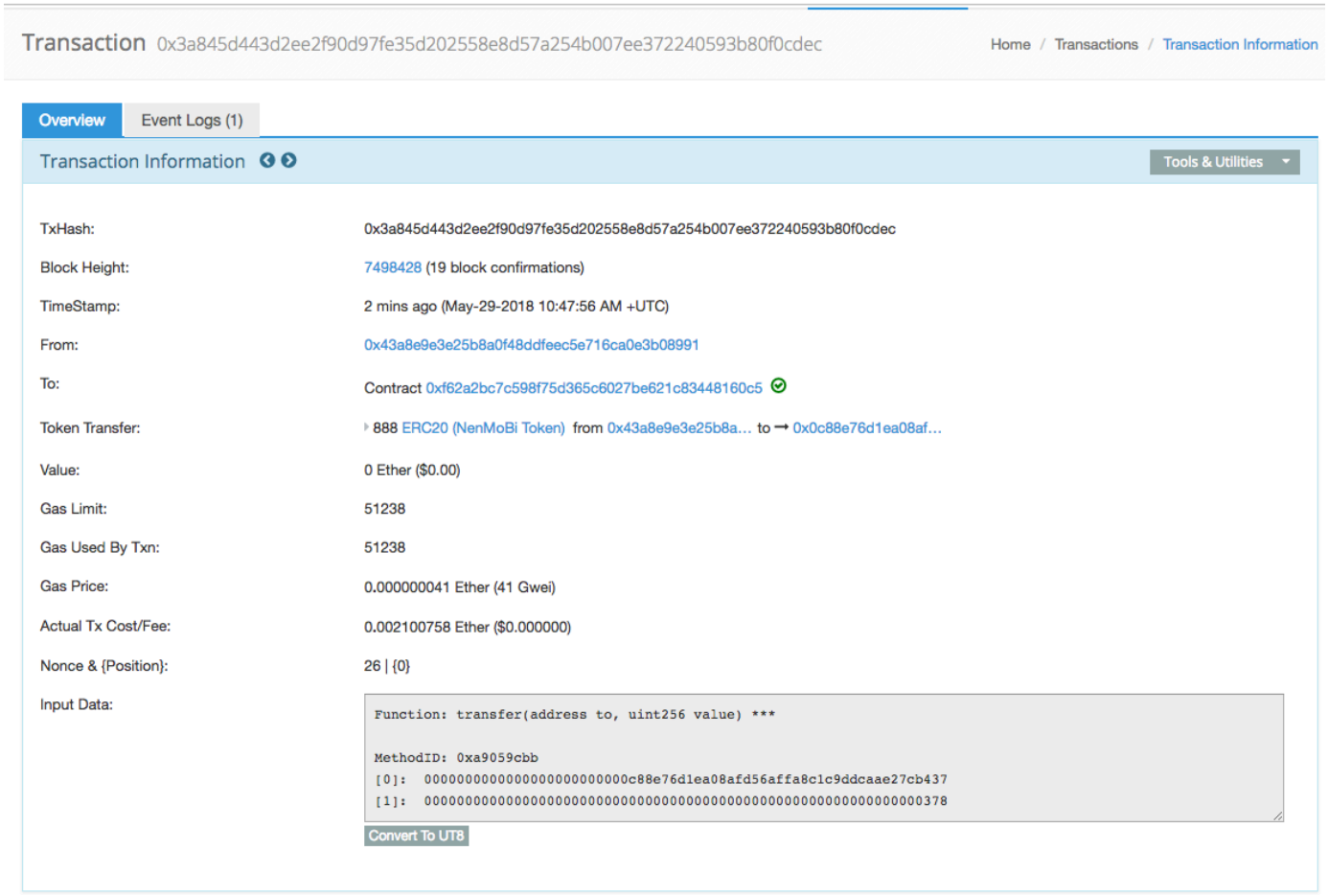
点击如图的"确认"按钮，会弹出MetaMask确认交易。



点击"Submti"确认提交交易，交易成功后网页显示了如下信息：



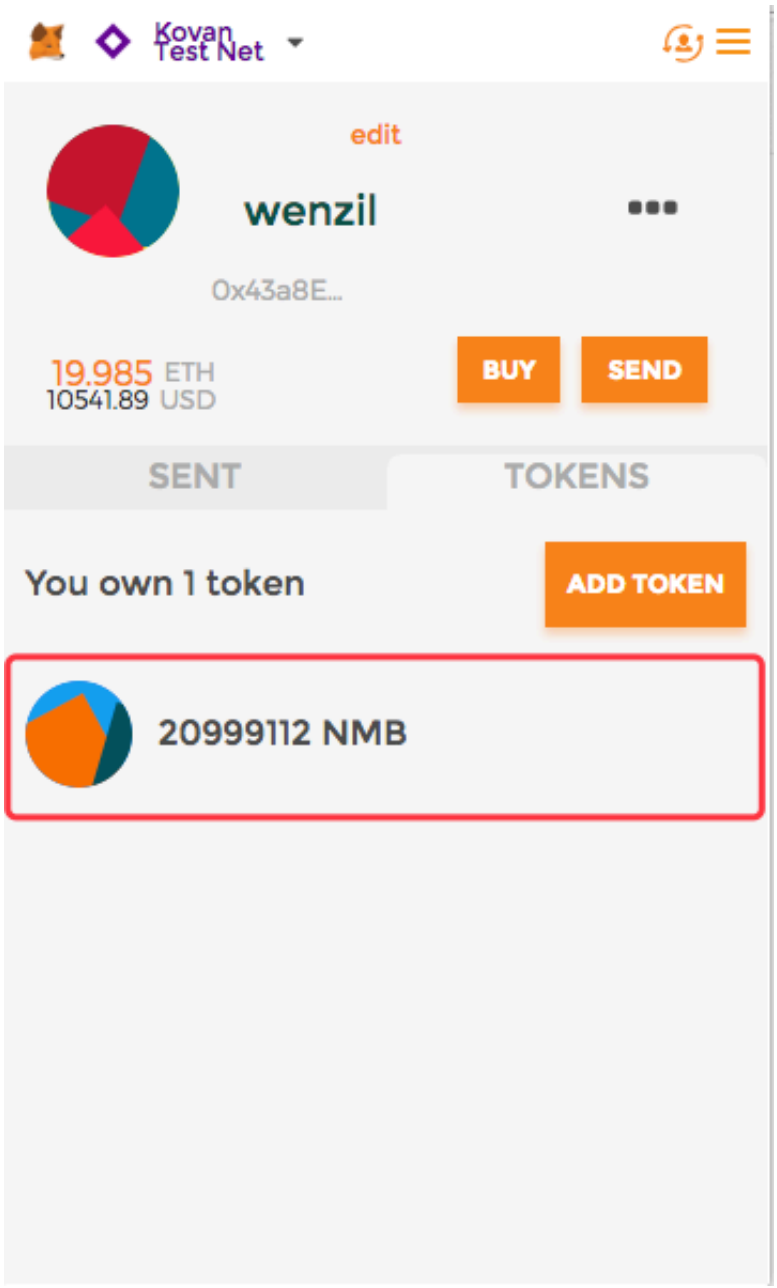
可以点击页面左下角的两个按钮来查看交易状态和交易信息。

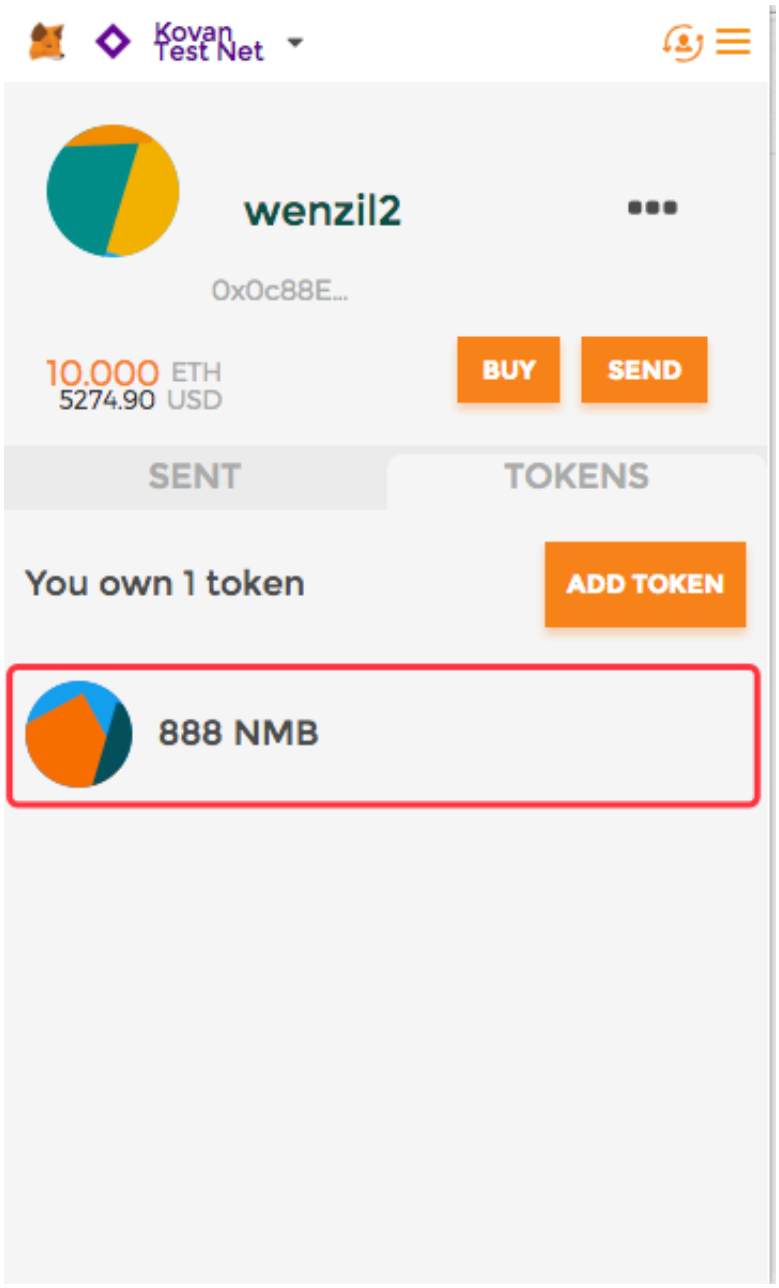


对应的交易状态地址：
https://www.myetherwallet.com/?txHash=0x3a845d443d2ee2f90d97fe35d202558e8d57a254b007ee372240593b80f0cdec#check-tx-status

对应的交易信息地址：
https://kovan.etherscan.io/tx/0x3a845d443d2ee2f90d97fe35d202558e8d57a254b007ee372240593b80f0cdec

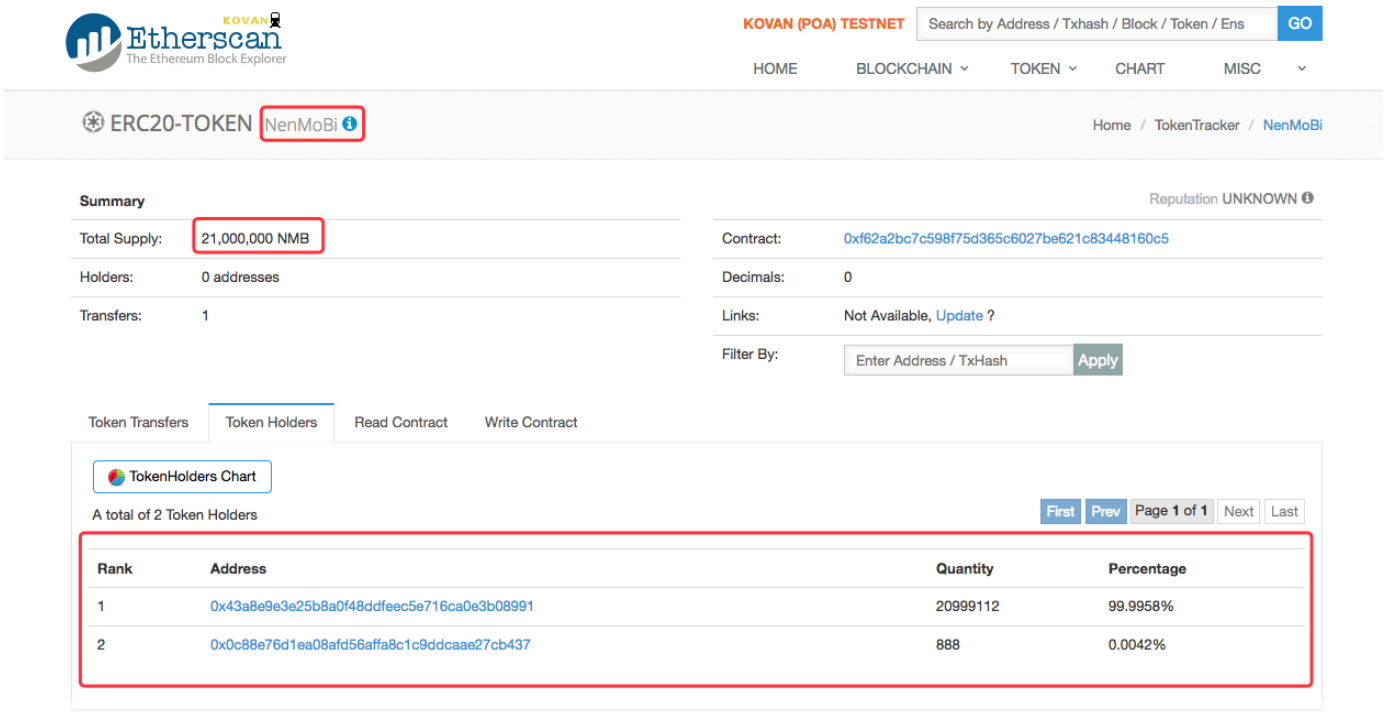
最后再查看两个账号对应的余额信息：


Kovan Test Net
edit
wenzil
0x43a8E...
19.985 ETH
10541.89 USD
BUY
SEND
SENT
TOKENS
You own 1 token
ADD TOKEN
20999112 NMB



账号1成功向账号2转了888个NMB（嫩模币）。

再次打开Etherscan查看该代币信息，发现多了一个交易信息，账号拥有者有两个。



搞定，收工。。。

拓展阅读：

不得不备的工具 - Etherscan