以太坊开发实战

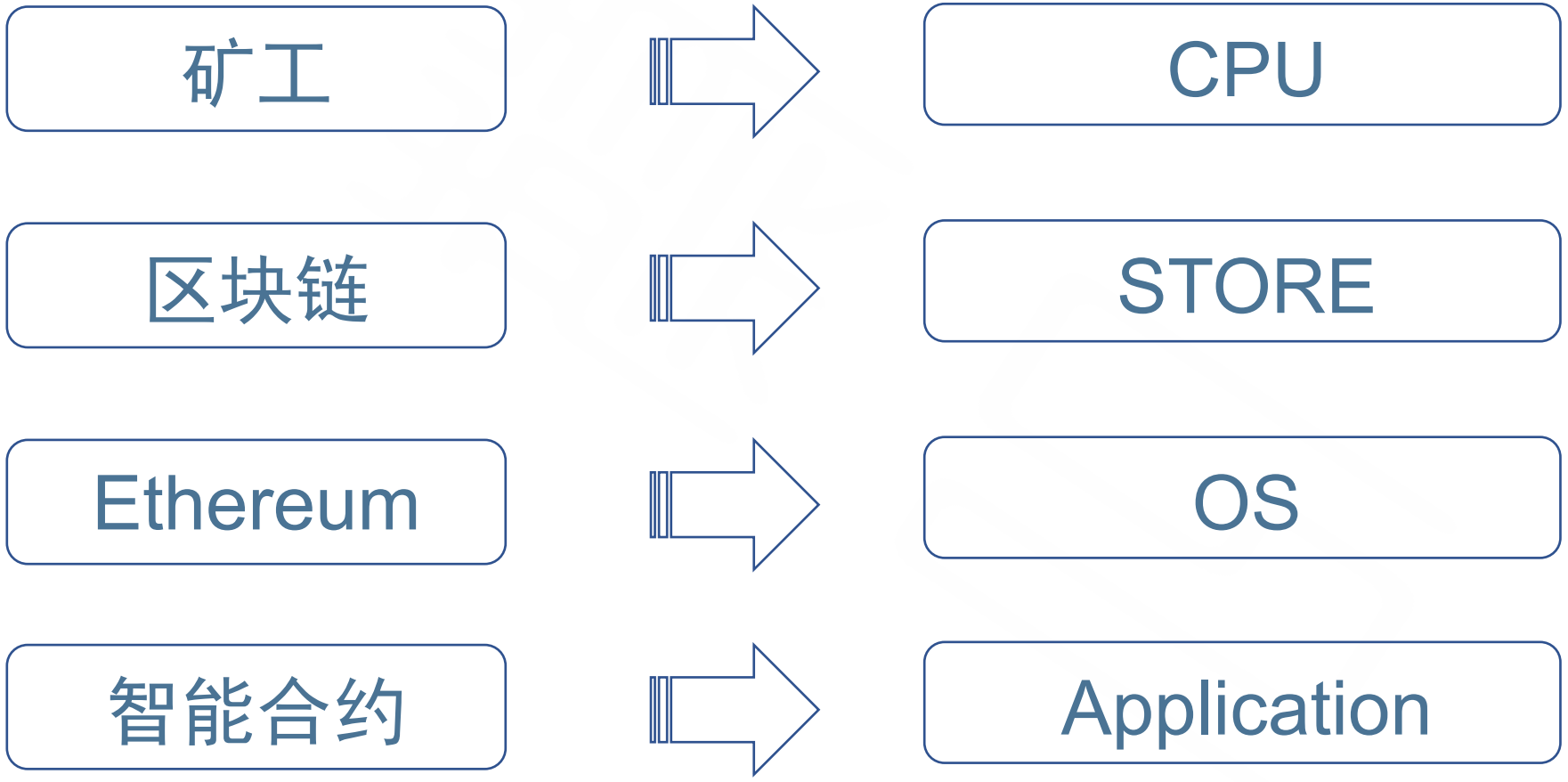
蔡一@志顶科技

目录

什么是以太坊

- 史上第一台“世界电脑”，任何人都可以使用的去中心化网络
- 可运行各种应用，应用被称为“智能合约”
- 不会受到设备当机、接收检查或被欺骗等的影响
- 以太坊拥有其本身的货币“以太币”
- 用户通过支付以太币来使用去中心化网络以执行所需完成的任务
- 任何人都可自愿的增加算力，并可赚取以太坊（挖矿）

“世界电脑”



智能合约

定义

- 智能合约是 1990s 年代由尼克萨博提出的理念，几乎与互联网同龄
- 智能合约-包含价值而且只有满足某些条件才能打开的加密箱子

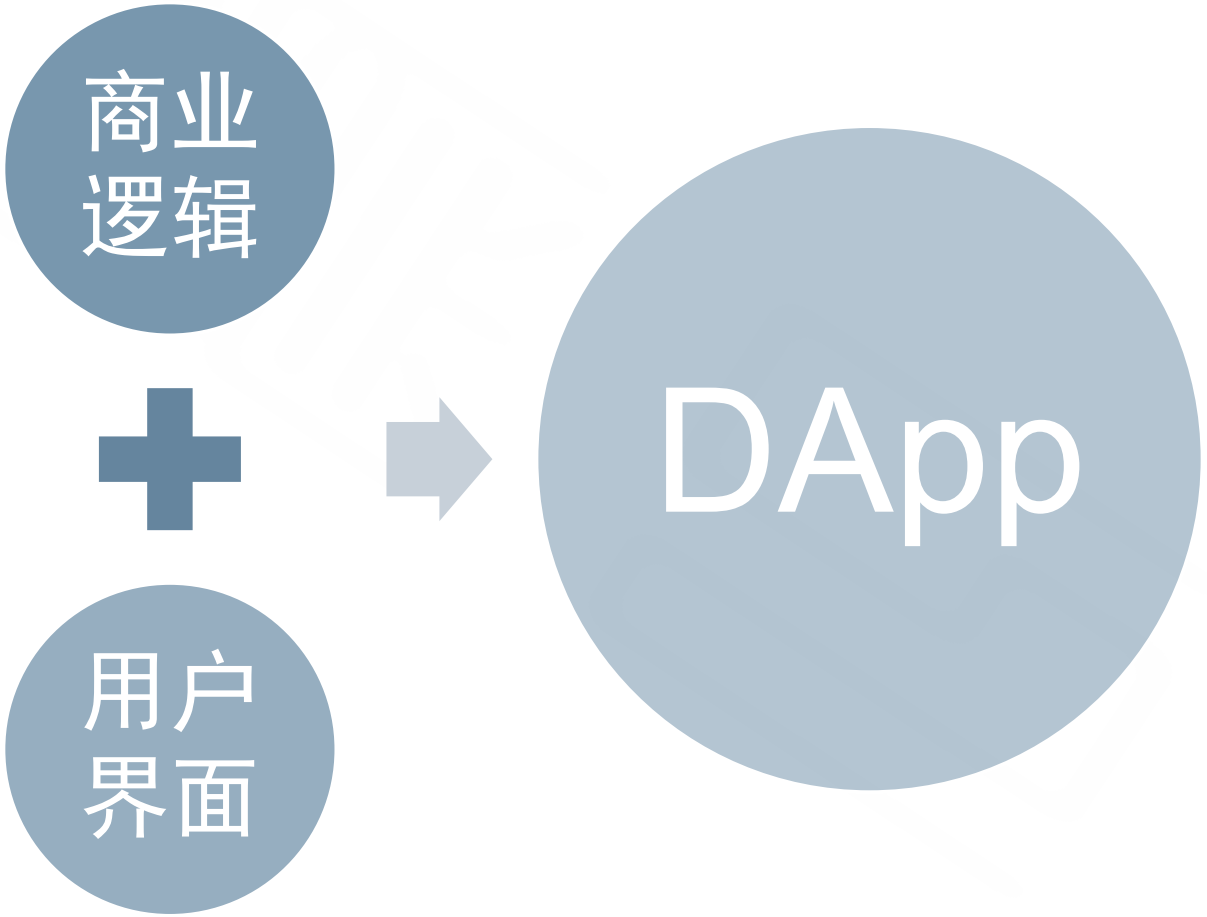
行为

- 智能合约可以接受和存储以太币、数据、或两者组合。
- 智能合约维持自己的状态，控制自己的资产和对接收到的外界信息或资产进行回应。

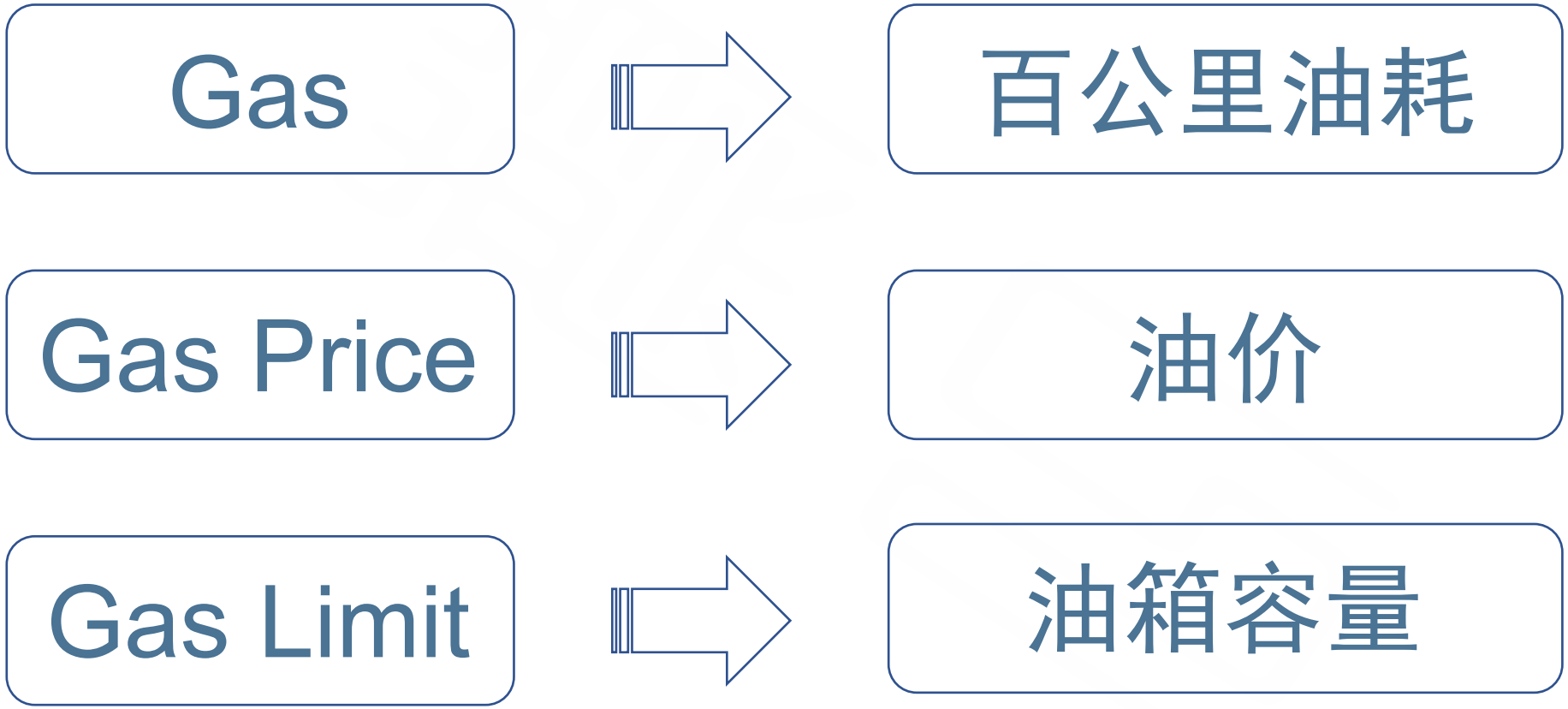
实现

- 支持图灵完备的脚本语言，允许开发者在上面开发任意应用
- Solidity是现在以太坊上最流行的智能合约语言。

DApp（分布式应用程序）



Gas



账户模型，而不是UTXO

优点

- ✓ 节省大量空间
- ✓ 可替代性强
- ✓ 简单
- ✓ 轻客户端

缺点

- ✗ 性能不如UTXO
- ✗ 防重放攻击实现复杂

目录

以太坊的起源

2013年

- Vitalik加入比特币的转型工作（Bitcoin 2.0），发现比特币在先天设计上无法突破的局限
- 他向比特币社群提出自己设想的程序，并想要融入现有的比特币区块链系统中，却被拒绝

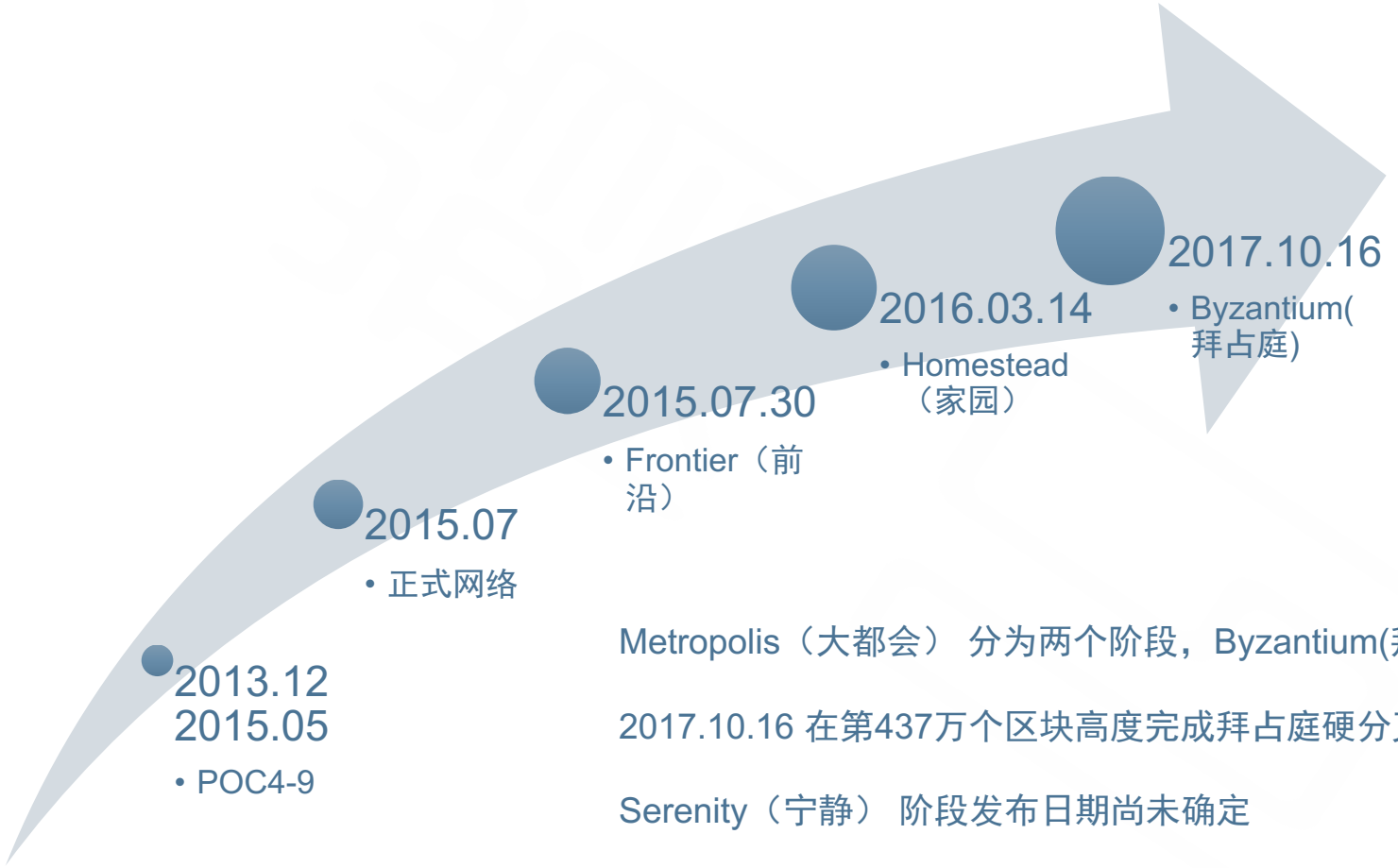
2013年年末

- 发布了以太坊初版白皮书，在全球的密码学货币社区陆续召集到一批认可以太坊理念的开发者，启动了项目

2014年7月

- 「以太坊计划」启动以太币众筹募资(ICO)，募得3万1千枚比特币（1840万美元）

以太坊路线图



Metropolis（大都会） 分为两个阶段，Byzantium(拜占庭)、Constantinople(君士坦丁堡)

2017.10.16 在第437万个区块高度完成拜占庭硬分叉

Serenity（宁静） 阶段发布日期尚未确定

目录

以太坊钱包种类

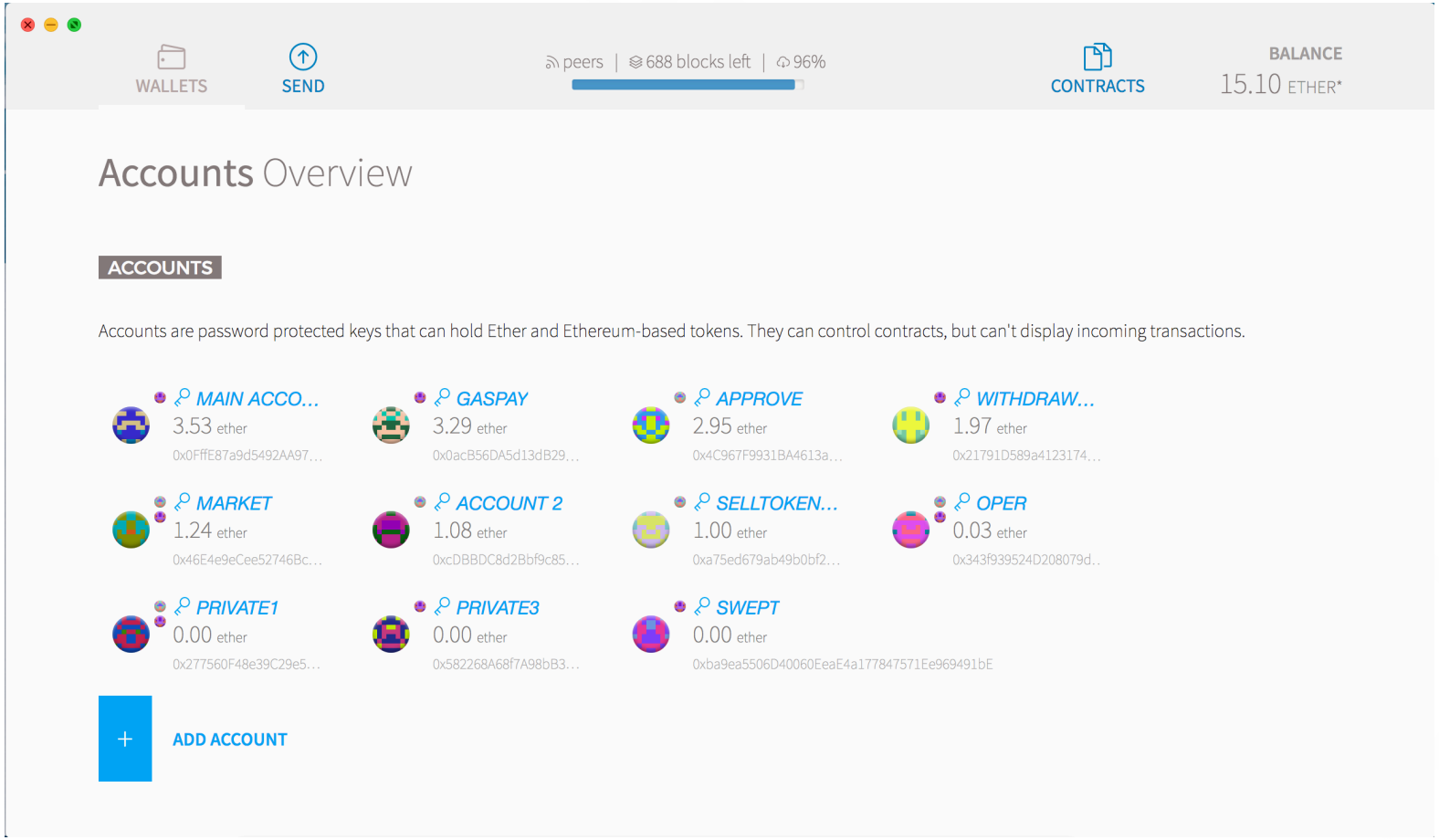
全节点钱包

- Ethereum Wallet
- Mist
- Parity

轻钱包

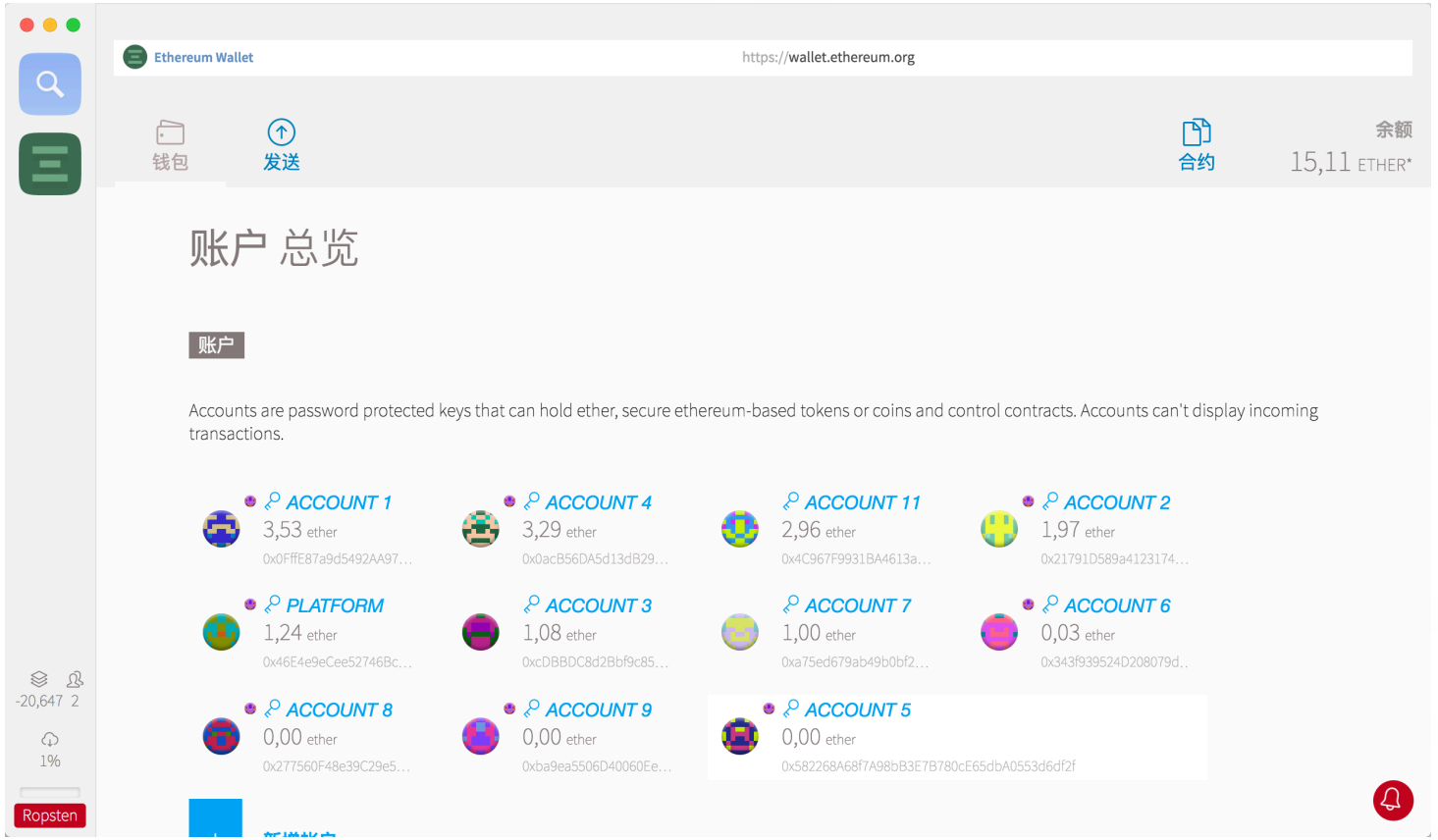
- MetaMask
- imToken

Ethereum Wallet



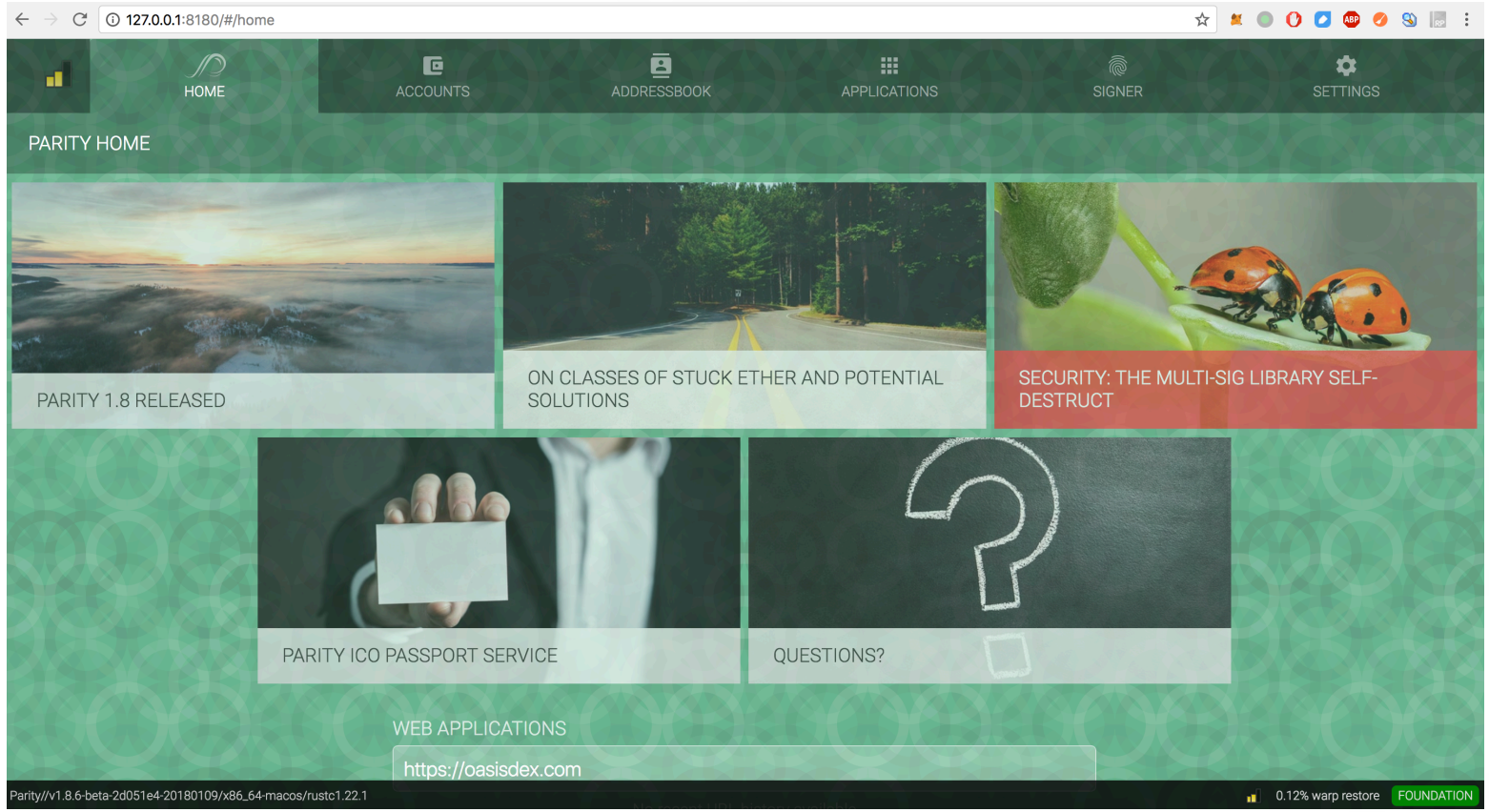
- ◇官方钱包
- ◇性能一般
- ◇同步慢，占用空间大
- ◇支持合约部署
- ◇支持合约函数执行

Mist



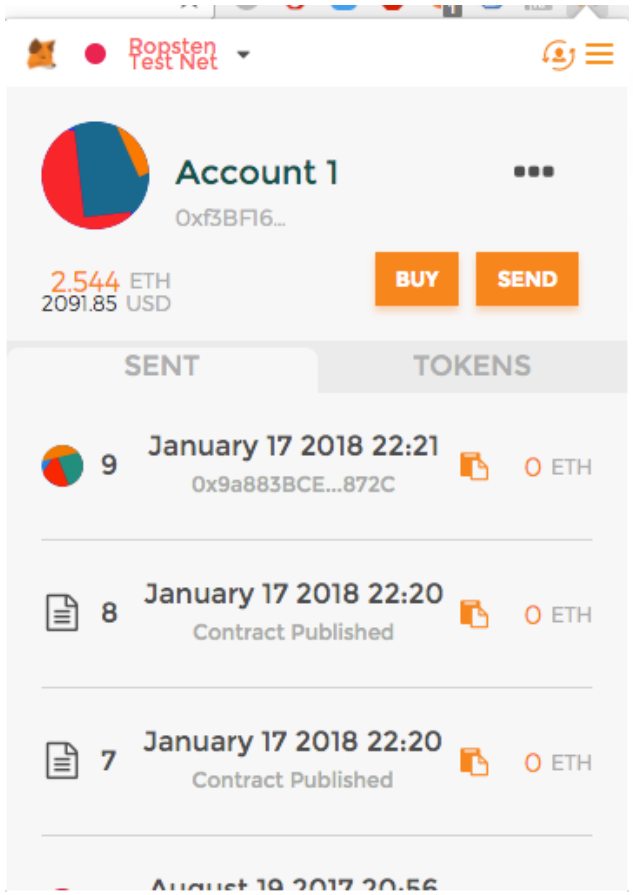
- ◇ DApp浏览器
- ◇ 内置Ethereum Wallet DApp
- ◇ 同步慢，占用空间大
- ◇ 支持合约部署
- ◇ 支持合约函数执行

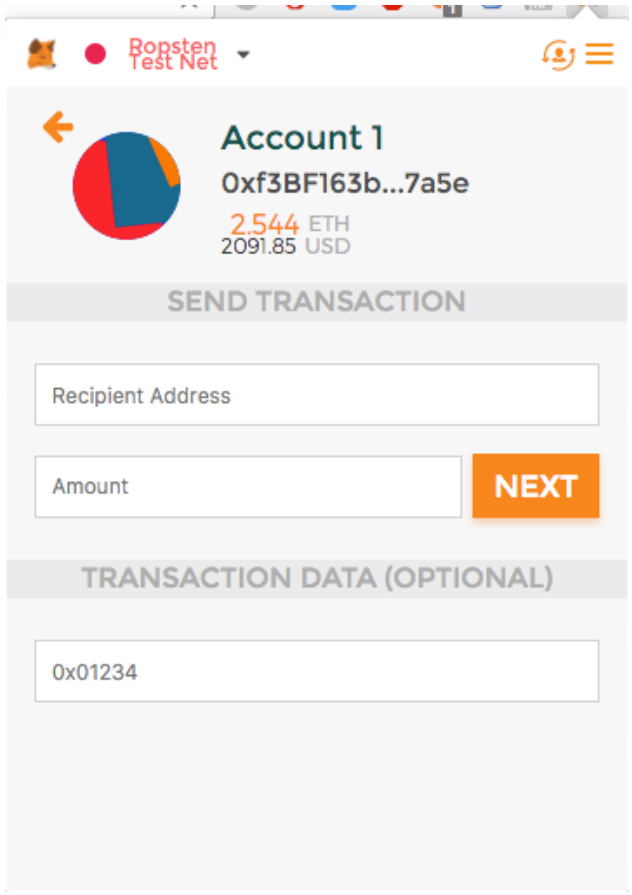
Parity

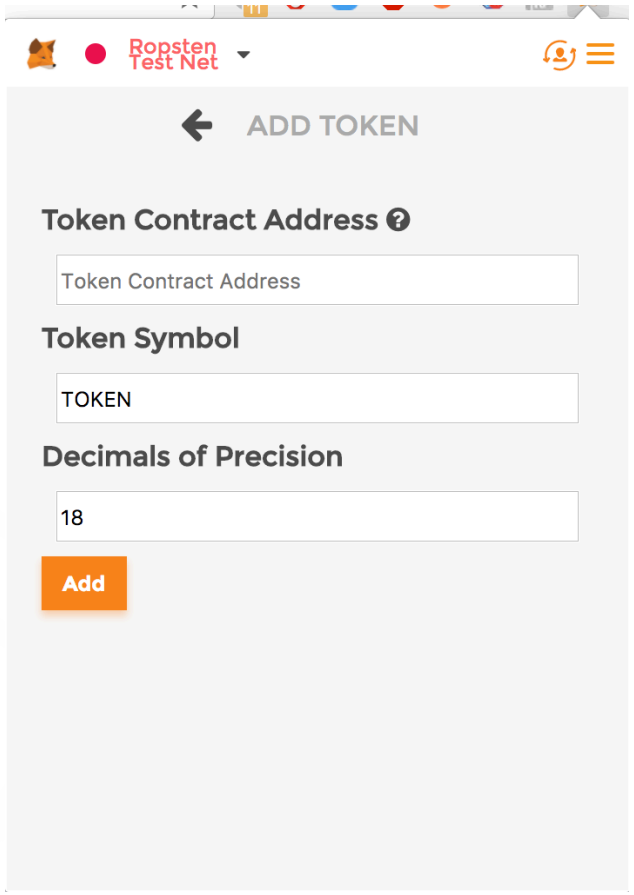


- ◇ 性能更好
- ◇ 占用系统的资源更少
- ◇ 同步很快，占用空间小
- ◇ 支持合约部署
- ◇ 支持定时发送交易

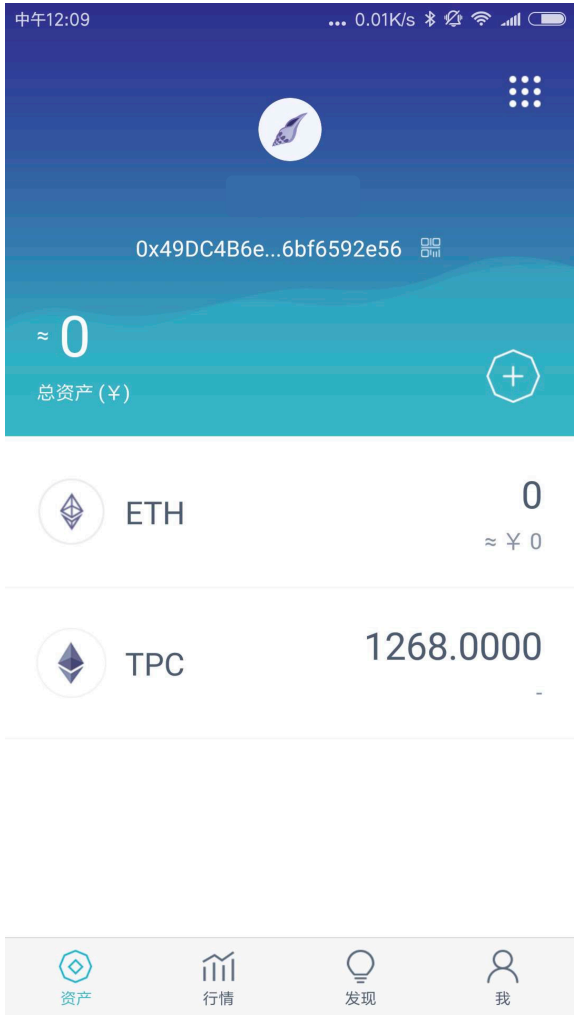
MetaMask

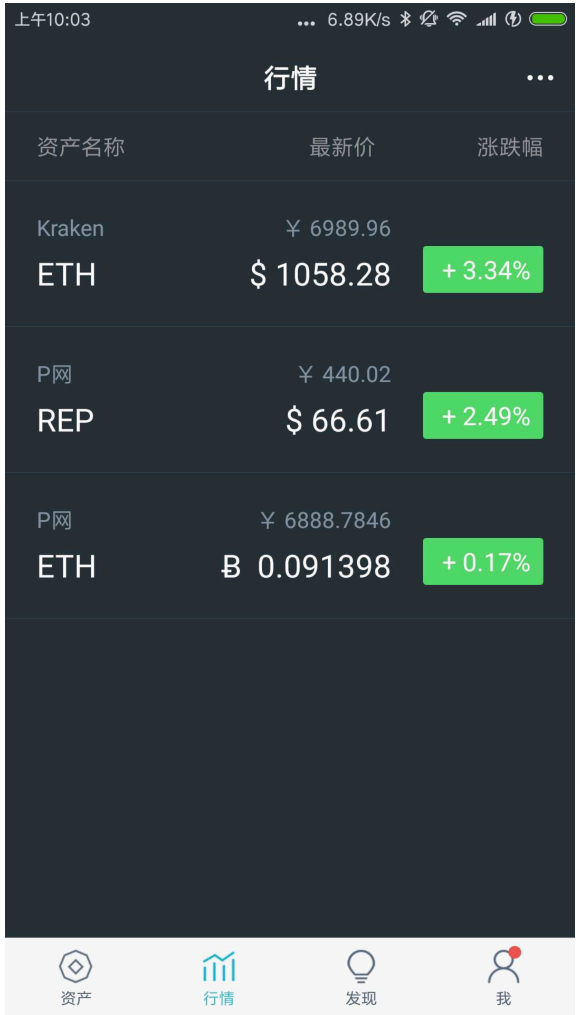

Ropsten Test Net
Account 1
0xf3BF16...
2.544 ETH
2091.85 USD
BUY
SEND
SENT
TOKENS
9
January 17 2018 22:21
0x9a883BCE...872C
0 ETH
8
January 17 2018 22:20
Contract Published
0 ETH
7
January 17 2018 22:20
Contract Published
0 ETH


Ropsten Test Net
Account 1
0xf3BF163b...7a5e
2.544 ETH
2091.85 USD
SEND TRANSACTION
Recipient Address
Amount
NEXT
TRANSACTION DATA (OPTIONAL)
0x01234


Ropsten Test Net
ADD TOKEN
Token Contract Address
Token Contract Address
Token Symbol
TOKEN
Decimals of Precision
18
Add

imToken


中午12:09
0.01K/s
0x49DC4B6e...6bf6592e56
≈ 0
总资产 (¥)
ETH
0
≈ ¥ 0
TPC
1268.0000
-
资产
行情
发现
我


上午10:03
6.89K/s
行情
资产名称
最新价
涨跌幅
Kraken
ETH
¥ 6989.96
$ 1058.28
+ 3.34%
P网
REP
¥ 440.02
$ 66.61
+ 2.49%
P网
ETH
¥ 6888.7846
Ƀ 0.091398
+ 0.17%
资产
行情
发现
我


中午12:09
4.19K/s
TPC 转账
收款人钱包地址
转账金额
备注
矿工费用
慢
0.00489600 ether
快
高级选项
下一步

密钥、地址、密码及助记词

私钥 → 公钥 → 地址

地址 = 银行卡号

密码 = 银行卡密码

私钥 = 银行卡号 + 银行卡密码

助记词 = 银行卡号 + 银行卡密码

Keystore + 密码 = 银行卡号 + 银行卡密码

目录

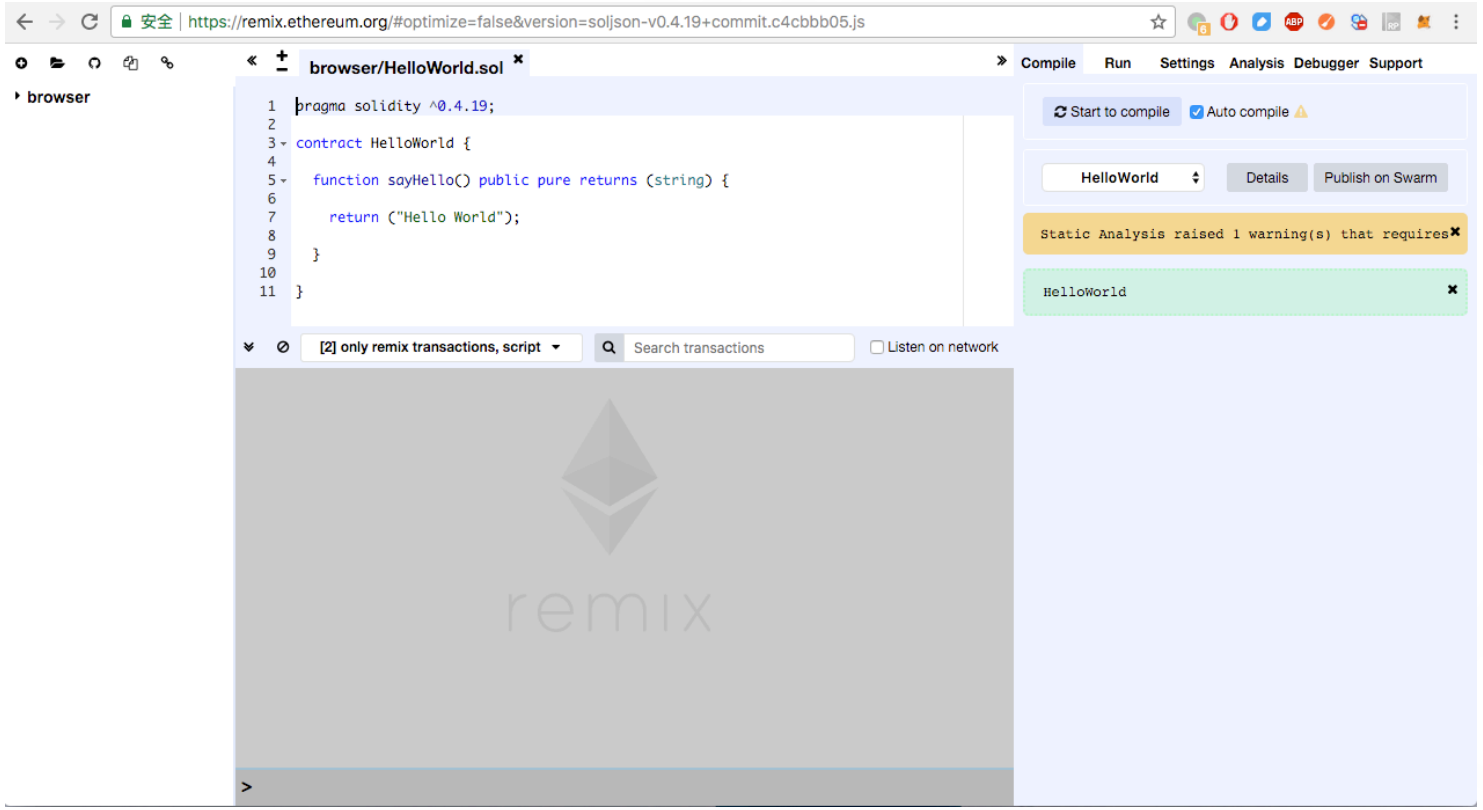
Solidity智能合约开发

- ☞ Solidity IDE
- ☐ Solidity初探—HelloWorld
- ☐ Solidity入门—Pet Shop
- ☐ Solidity进阶—ERC20

Solidity IDE

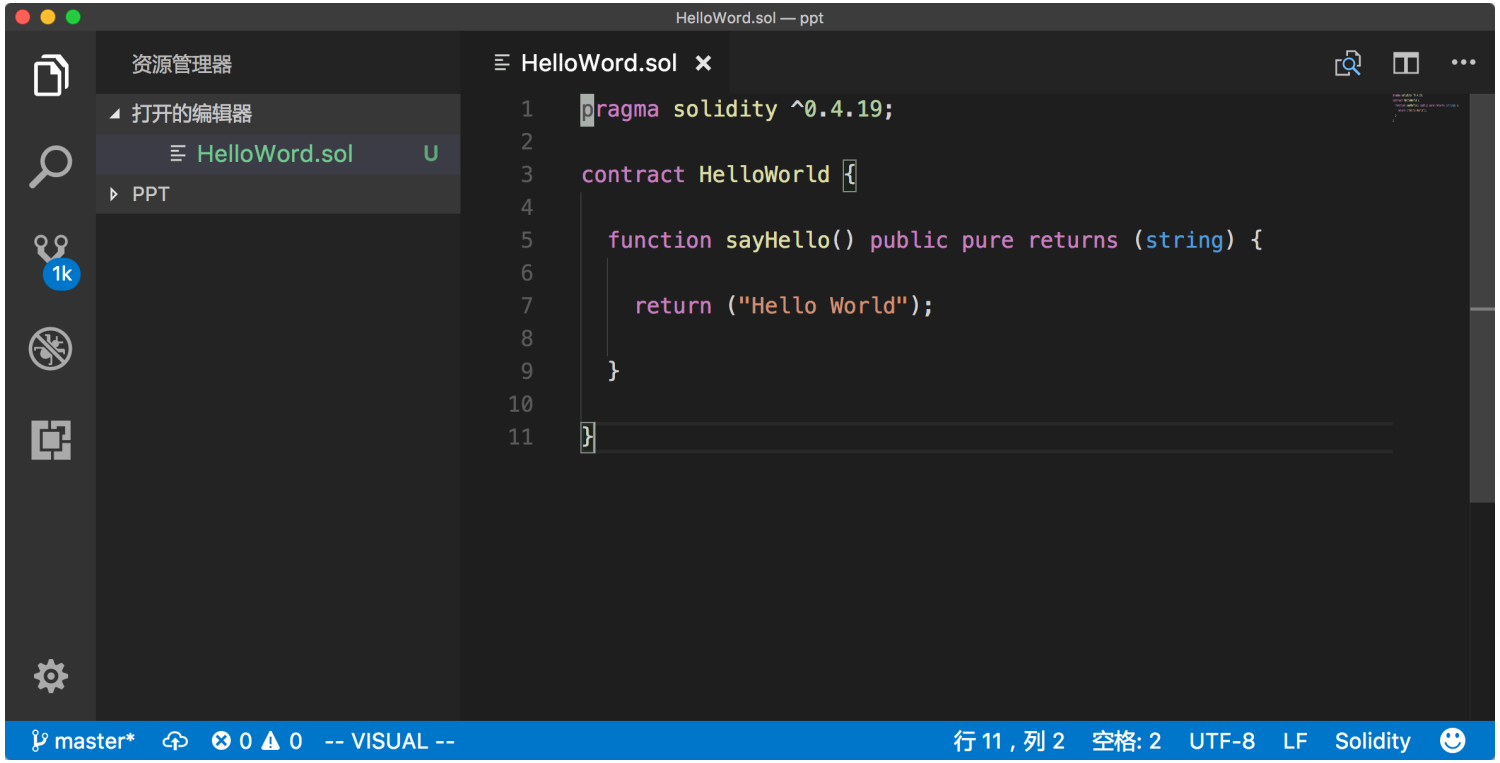
- Remix
- Visual Studio Code
- Atom
- Intellij IDEA

remix



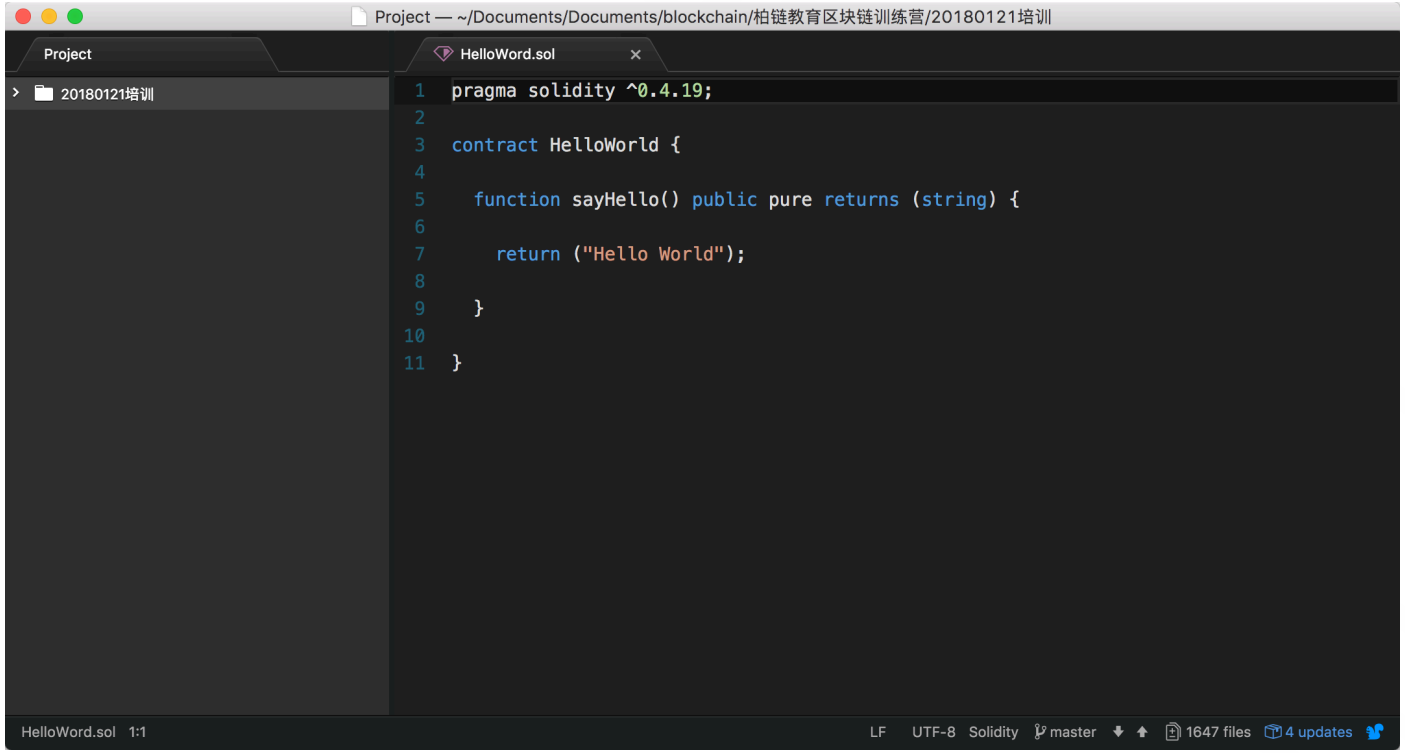
- ◇ 基于浏览器，无需安装
- ◇ 语法高亮
- ◇ 支持测试和部署
- ◇ 自带JavaScript实现的EVM
- ◇ 可与MetaMask结合
- ◇ 可自定义钱包服务节点

Visual Studio Code



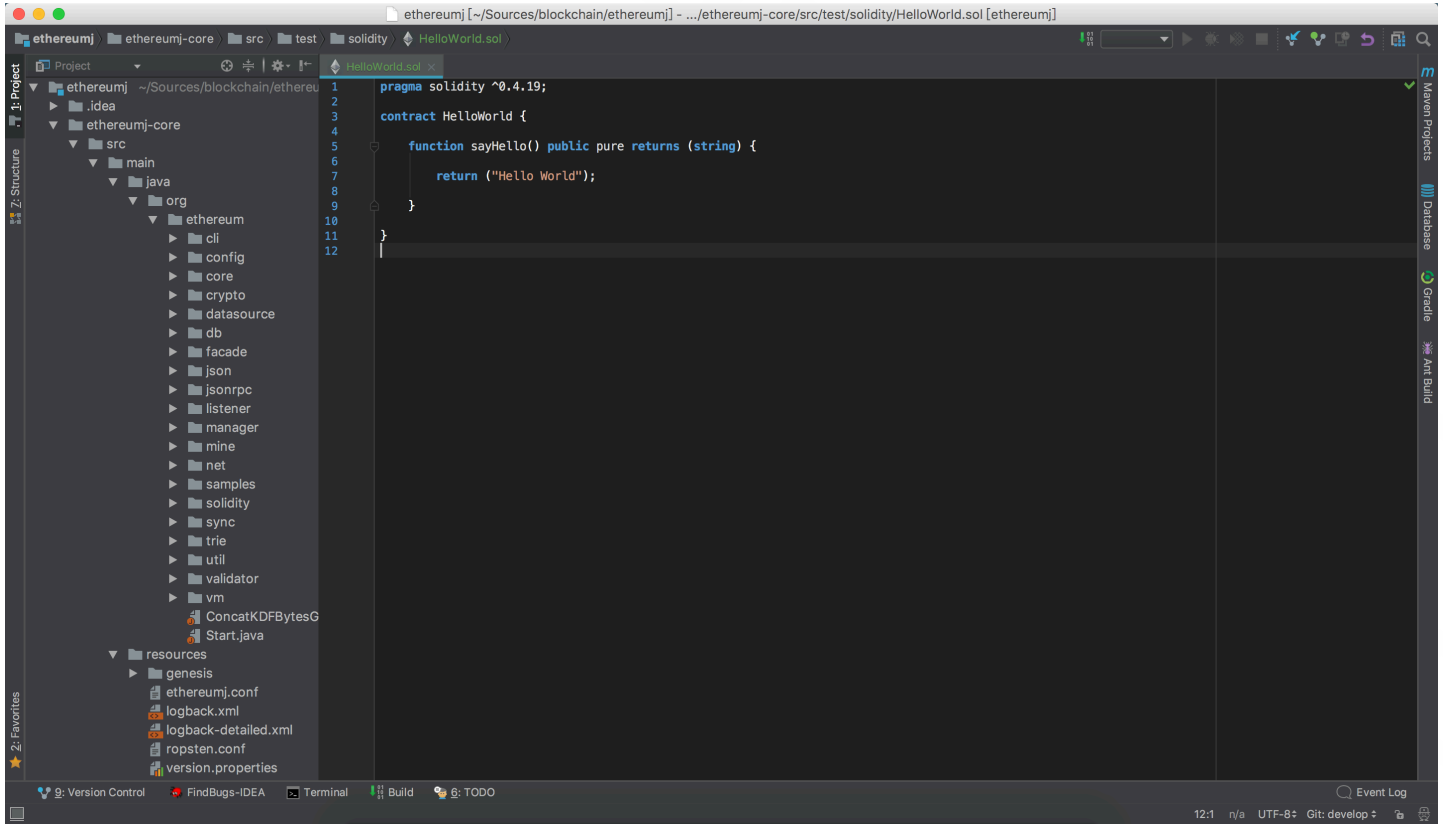
- ◇ 全功能Editor
- ◇ 配合Solidity插件使用
- ◇ 语法高亮
- ◇ 支持自动完成
- ◇ 支持自动编译
- ◇ 不能测试和部署

Atom



- ◇ 和 Visual Studio Code类似
- ◇ 配合Solidity插件使用
- ◇ 语法高亮
- ◇ 支持自动完成
- ◇ 支持自动编译
- ◇ 不能测试和部署

Intellij IDEA



- ◇ 重量级IDE
- ◇ 适合混合语言项目
- ◇ 仅支持语法高亮
- ◇ 不支持编译
- ◇ 不能测试和部署

Solidity智能合约开发

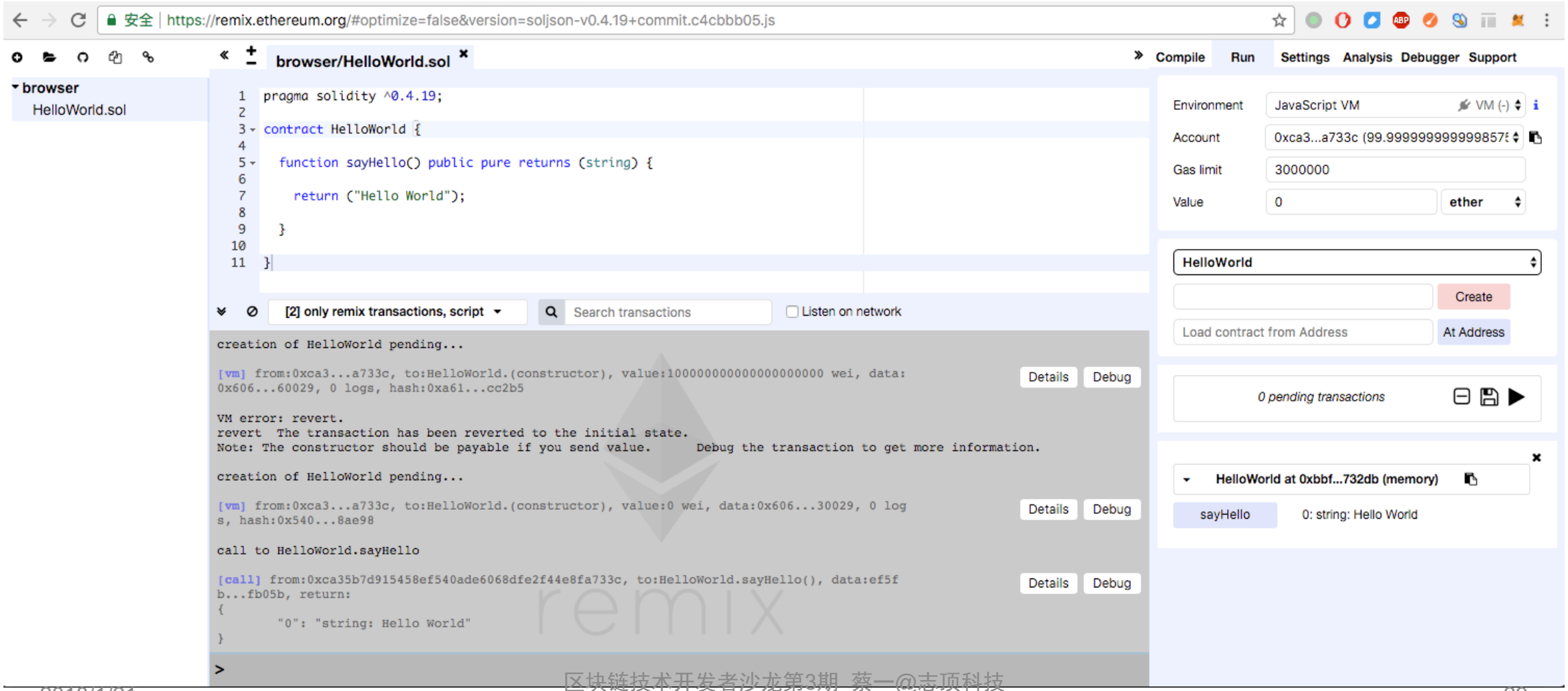
- ☐ Solidity IDE
- ☞ Solidity初探—HelloWorld
- ☐ Solidity入门—Pet Shop
- ☐ Solidity进阶—ERC20
- ☐ 开发安全的智能合约—OpenZepplin

HelloWorld - Code

```
pragma solidity ^0.4.19;

contract HelloWorld {

  function sayHello() public pure returns (string) {

    return ("Hello World");

  }

}
```

目前使用的solidity版本，不同版本的solidity可能会编译出不同的bytecode。^代表兼容solidity0.4.19以后的版本（不包括0.5.0）。

HelloWorld - Run



Solidity智能合约开发

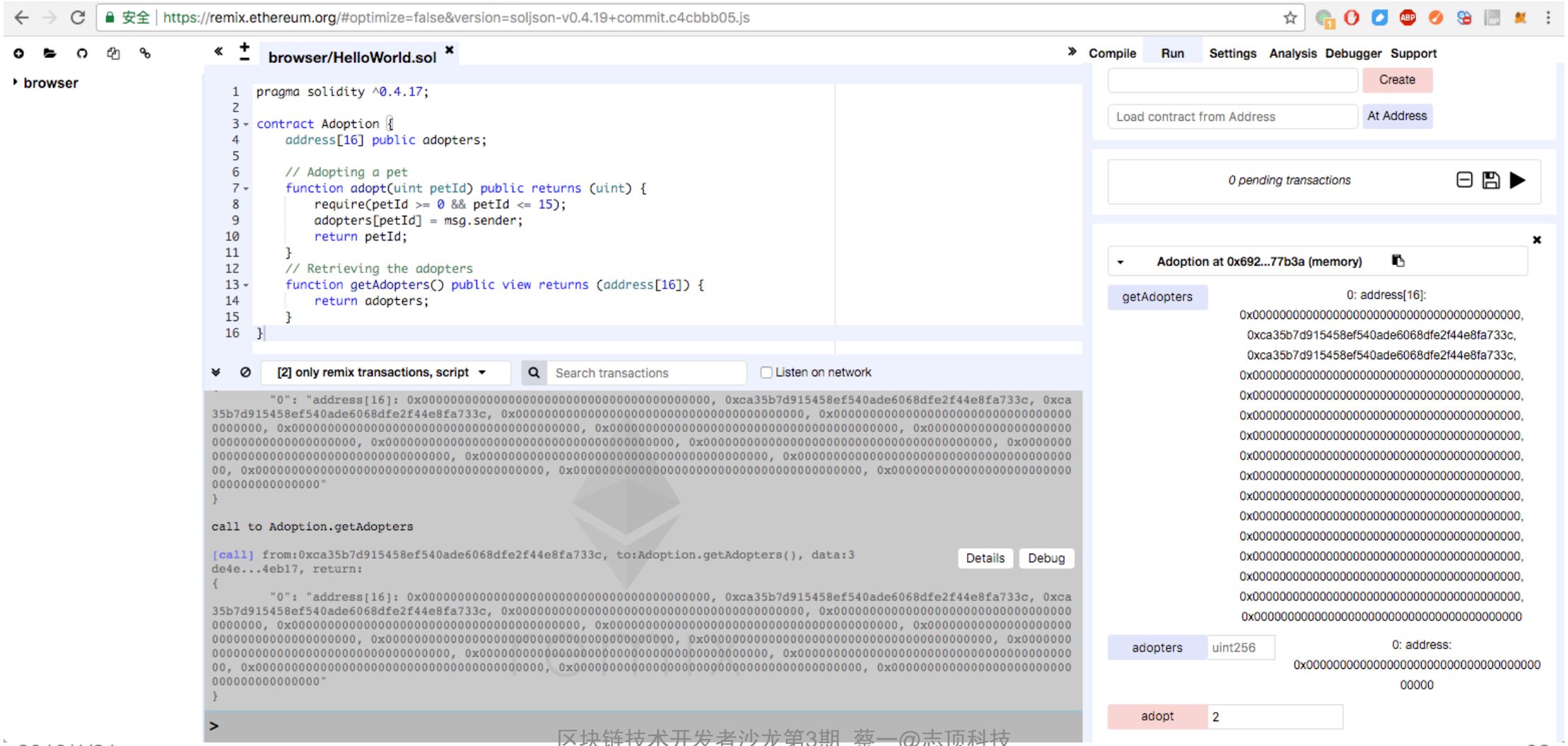
- ☐ Solidity IDE
- ☐ Solidity初探—HelloWorld
- ☞ Solidity入门—Pet Shop
- ☐ Solidity进阶—ERC20
- ☐ 开发安全的智能合约—OpenZepplin

Pet Shop - Code

```
pragma solidity ^0.4.17;

contract Adoption {
    address[16] public adopters;

    // Adopting a pet
    function adopt(uint petId) public returns (uint) {
       require(petId >= 0 && petId <= 15);
        adopters[petId] = msg.sender;
       return petId;
    }
    // Retrieving the adopters
    function getAdopters() public view returns (address[16]) {
        return adopters;
    }
}
```

From http://truffleframework.com/tutorials/pet-shop

Pet Shop - Run


安全 | https://remix.ethereum.org/#optimize=false&version=soljson-v0.4.19+commit.c4cbbb05.js
browser
browser/HelloWorld.sol
Compile
Run
Settings
Analysis
Debugger
Support
Create
Load contract from Address
At Address
0 pending transactions
pragma solidity ^0.4.17;
contract Adoption {
address[16] public adopters;
// Adopting a pet
function adopt(uint petId) public returns (uint) {
require(petId >= 0 && petId <= 15);
adopters[petId] = msg.sender;
return petId;
}
// Retrieving the adopters
function getAdopters() public view returns (address[16]) {
return adopters;
}
}
[2] only remix transactions, script
Search transactions
Listen on network
call to Adoption.getAdopters
Details
Debug
Adoption at 0x692...77b3a (memory)
getAdopters
adopters
uint256
adopt
2

Solidity智能合约开发

- ☐ Solidity IDE
- ☐ Solidity初探—HelloWorld
- ☐ Solidity入门—Pet Shop
- ☞ Solidity进阶—ERC20
- ☐ 开发安全的智能合约—OpenZepplin

ERC20规范

ERC20规范定义了代币（Token）的标准接口。早期为Ethereum Request for Comments (ERC), 9月4日正式加入EIP，变为EIP20。

```
function name() view returns (string name);
function symbol() view returns (string symbol);
function decimals() view returns (uint8 decimals);
function totalSupply() view returns (uint256 totalSupply);
function balanceOf(address _owner) view returns (uint256 balance);
function transfer(address _to, uint256 _value) returns (bool success);
function transferFrom(address _from, address _to, uint256 _value) returns (bool
success);
function approve(address _spender, uint256 _value) returns (bool success);
function allowance(address _owner, address _spender) view  returns (uint256
remaining);
```

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

ERC721规范

```
function name() constant returns (string name);
function symbol() constant returns (string symbol);
function totalSupply() constant returns (uint256 totalSupply);
function balanceOf(address _owner) constant returns (uint256 balance);
function ownerOf(uint256 _tokenId) constant returns (address owner);
function approve(address _to, uint256 _tokenId);
function takeOwnership(uint256 _tokenId);
function transfer(address _to, uint256 _tokenId);
function tokenOfOwnerByIndex(address _owner, uint256 _index) constant
returns (uint tokenId);
function tokenMetadata(uint256 _tokenId) constant returns (string infoUrl);
```

Solidity智能合约开发

- ☐ Solidity IDE
- ☐ Solidity初探—HelloWorld
- ☐ Solidity入门—Pet Shop
- ☐ Solidity进阶—ERC20 Token
- ☞ 开发安全的智能合约—OpenZepplin

安全事件—The DAO

2016年6月17日，由于编写的智能合约存在着重大缺陷，区块链业界最大的众筹项目TheDAO(被攻击前拥有1亿美元左右资产)遭到攻击，导致300多万以太币资产被分离出TheDAO 资产池。

```
function splitDAO( uint _proposalID, address _newCurator) noEther onlyTokenholders returns (bool _success) {
    ... ...
    uint fundsToBeMoved =
        (balances[msg.sender] * p.splitData[0].splitBalance) / p.splitData[0].totalSupply;

    if (p.splitData[0].newDAO.createTokenProxy.value(fundsToBeMoved)(msg.sender) == false)
            throw;
    ... ...
    // Burn DAO Tokens
    Transfer(msg.sender, 0, balances[msg.sender]);
    withdrawRewardFor(msg.sender); // be nice, and get his rewards
    totalSupply -= balances[msg.sender];
    balances[msg.sender] = 0;
    paidOut[msg.sender] = 0;
    return true;
}
function withdrawRewardFor(address _account) noEther internal returns (bool _success) {
    if ((balanceOf(_account) * rewardAccount.accumulatedInput()) / totalSupply < paidOut[_account])
        throw;

    uint reward =
        (balanceOf(_account) * rewardAccount.accumulatedInput()) / totalSupply - paidOut[_account];
    if (!rewardAccount.payOut(_account, reward))
        throw;
    paidOut[_account] += reward;
    return true;
}
function payOut(address _recipient, uint _amount) returns (bool) {
        if (msg.sender != owner || msg.value > 0 || (payOwnerOnly && _recipient != owner))
            throw;
        if (_recipient.call.value(_amount)()) {
            PayOut(_recipient, _amount);
            return true;
        } else {
            return false;
        }
```

漏洞一：

Race To Empty

漏洞二：

Gas Not Set

安全事件—Parity多签名合约漏洞

2017年11月7日Parity因为合约中的一个新bug再次被黑客攻击，致使大约50万枚以太币被锁在多签智能合约里面，当时价值大约1.5亿美元，包括Gavin Woods作为创始人的新项目Polkadot公募获得的9000万美元。

```
// constructor - just pass on the owner array to the multiowned and
// the limit to daylimit
function initWallet(address[] _owners, uint _required, uint _daylimit) only_uninitialized {
  initDaylimit(_daylimit);
  initMultiowned(_owners, _required);
}

// constructor is given number of sigs required to do protected "onlymanyowners" transactions
// as well as the selection of addresses capable of confirming them.
function initMultiowned(address[] _owners, uint _required) only_uninitialized {
  m_numOwners = _owners.length + 1;
  m_owners[1] = uint(msg.sender);
  m_ownerIndex[uint(msg.sender)] = 1;
  for (uint i = 0; i < _owners.length; ++i)
  {
    m_owners[2 + i] = uint(_owners[i]);
    m_ownerIndex[uint(_owners[i])] = 2 + i;
  }
  m_required = _required;
}
```

- 改了旧Bug，出了新Bug
- 逻辑不严密
- 测试不充分

https://github.com/paritytech/parity/issues/6995

如何才能开发安全的智能合约

ConsenSys: Ethereum Smart Contract Security Best Practices

https://consensys.github.io/smart-contract-best-practices/

OpenZepplin: zeppelin-solidity

https://github.com/OpenZeppelin/zeppelin-solidity

OpenZepplin

- 一种构建安全智能合约的开源架构
- 拉动式支付（Pull payment）辅助模块
- 合约生命周期工具
- 容错和自动挑错奖励
- 可重用的基础组件：代币发行、众筹等
- 探究形式验证理念
- 与Oracle更好的接口
- 更好的重用代码工具

- Ownable
  - Ownable( )
  - modifier onlyOwner( )
  - transferOwnership(address newOwner) onlyOwner
- Claimable
  - transfer(address newOwner) onlyOwner
  - modifier onlyPendingOwner
  - claimOwnership( ) onlyPendingOwner
- Migrations
  - upgrade(address new_address) onlyOwner
  - setCompleted(uint completed) onlyOwner**
- SafeMath
  - assert(bool assertion) internal
  - mul(uint256 a, uint256 b) internal returns (uint256)
  - sub(uint256 a, uint256 b) internal returns (uint256)
  - add(uint256 a, uint256 b) internal returns (uint256)
- LimitBalance
  - LimitBalance(unit _limit)
  - modifier limitedPayable()
- PullPayment
  - asyncSend(address dest, uint amount) internal
  - withdrawPayments( )
- StandardToken
  - approve(address _spender, uint _value) returns (bool success)
  - allowance(address _owner, address _spender) constant returns (uint
  - balanceOf(address _owner) constant returns (uint balance)
  - transferFrom(address _from, address _to, uint _value) returns (bool s
  - function transfer(address _to, uint _value) returns (bool success)

目录

DApp开发

- ☞ Dapp 基础
- ☐ 开发框架 Truffle
- ☐ 开发框架 Ganache

DApp基础

- JSON RPC API – 与节点交互的低级JSON RPC 2.0界面。这个API被Web3 JavaScript API使用。
- Web3 JavaScript API – 想要和以太坊节点交互的时候，主要用到的JavaScript SDK。
- Solidity – Solidity是以太坊开发的智能合约语言，编译到以太坊虚拟机操作码。
- HTML5，CSS，JavaScript – 用户界面开发
- 测试网络 – 测试网络帮助开发者开发和测试以太坊代码及网络互动，不需花费主网络上自己的以太币。

JSON RPC

JSON-RPC是一个无状态、轻量级的远程调用（RPC）协议，与传输协议无关，可使用Socket、HTTP或者其它协议，它使用JSON（RFC 4627）作为数据格式。

请求格式:

```
{
    "jsonrpc" : 2.0,
    "method" : "sayHello",
    "params" : ["Hello JSON-RPC"],
    "id" : 1
}
```

应答格式:

```
{
    "jsonrpc" : 2.0,
     "result" : "Hell JSON-RPC",
    "error" : null,
    "id" : 1
}
```

http://www.jsonrpc.org/specification

JSON-RPC API Reference

- web3_clientVersion
- web3_sha3
- net_version
- net_peerCount
- net_listening
- eth_protocolVersion
- eth_syncing
- eth_coinbase
- eth_mining
- eth_hashrate
- eth_gasPrice
- eth_accounts
- eth_blockNumber
- eth_getBalance
- db_putString
- db_getString
- db_putHex
- db_getHex
- shh_post
- shh_version
- shh_newIdentity
- shh_hasIdentity
- shh_newGroup
- shh_addToGroup
- shh_newFilter
- shh_uninstallFilter
- shh_getFilterChanges
- shh_getMessages

https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_sendtransaction

Web3 JavaScript API

- Ethereum JavaScript API
- 是对以太坊节点JSON RPC API的封装
- The web3-eth is for the ethereum blockchain and smart contracts
- The web3-shh is for the whisper protocol to communicate p2p and broadcast
- The web3-bzz is for the swarm protocol, the decentralized file storage
- The web3-utils contains useful helper functions for DApp developers.

https://web3js.readthedocs.io/en/1.0/web3-eth.html#gettransactionreceipt

Web3.js API Reference

- web3
  - version
    - api
    - node/getNode
    - network/getNetwork
    - ethereum/getEthereum
    - whisper/getWhisper
  - isConnected()
  - setProvider(provider)
  - currentProvider
  - reset()
  - sha3(string, options)
  - toHex(stringOrNumber)
  - toAscii(hexString)
  - fromAscii(textString, [padding])
  - db
    - putString(name, key, value)
    - getString(- var rXArray = stringToNumberA
    - putHex(name, key, value)
    - getHex(name, key)
  - shh
    - post(postObject)
    - newIdentity()
    - hasIdentity(hexString)
    - newGroup(_id, _who)
    - addToGroup(_id, _who)
    - filter(object/string)
      - watch(callback)
      - stopWatching(callback)
      - get(callback)

开发框架Truffle

Truffle是什么？

- Truffle是针对基于以太坊的Solidity语言的一套开发框架。本身基于Javascript。

学习Truffle前要知道什么？

- 基于Javascript的，知道基本语法、模块、Promise
- 了解Solidity
- 了解以太坊的基础知识

Truffle有什么用？

- 开发，测试，部署一行命令都可以搞定
- 提供了一套类似maven或gradle这样的项目构建机制，能自动生成相关目录
- 可在JavaScript中直接操作对应的合约函数
- 提供了控制台，使用框架构建后，可以直接在命令行调用输出结果，可极大方便开发调试
- 提供了监控合约，配置变化的自动发布，部署流程。不用每个修改后都重走整个流程

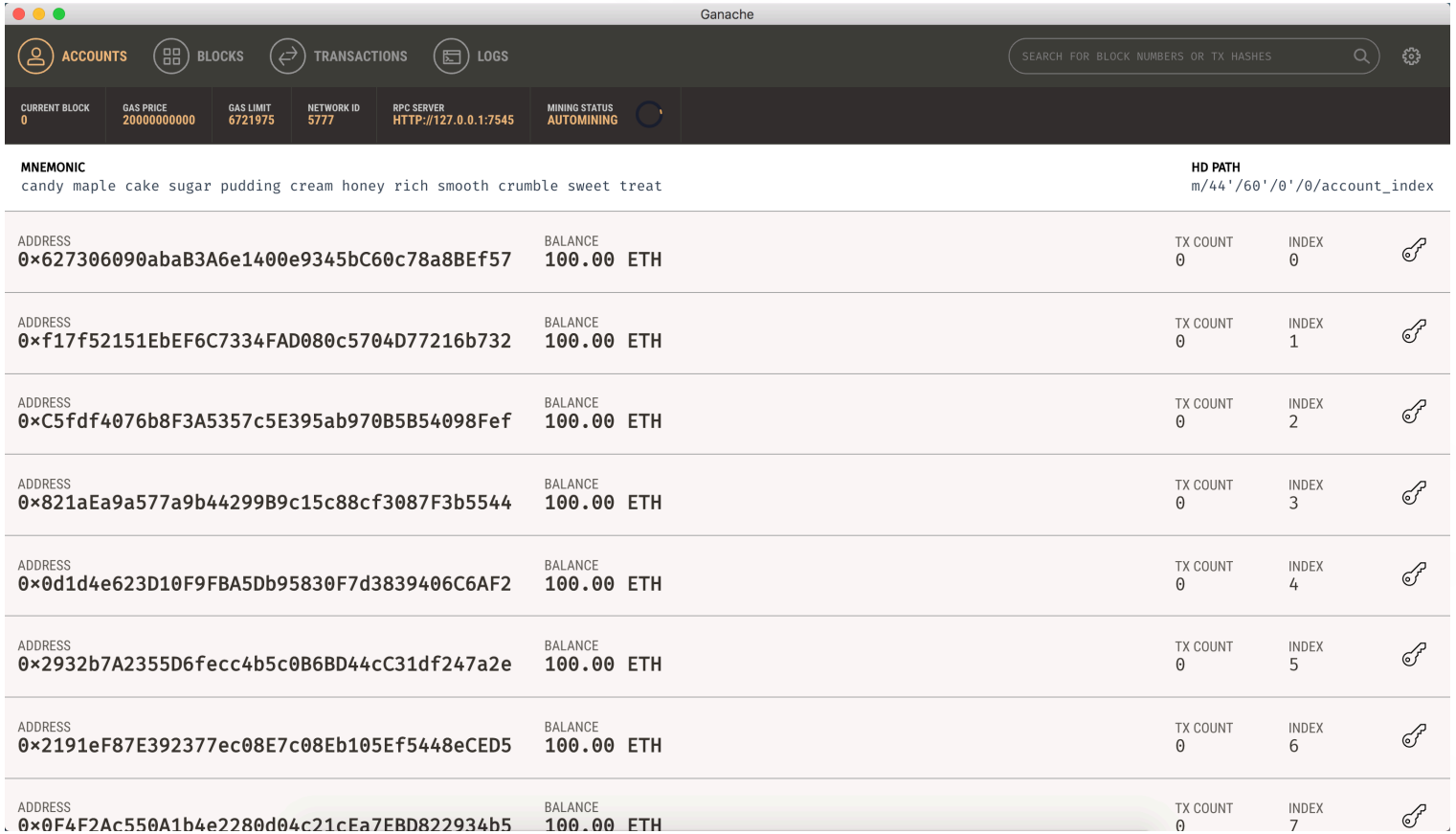
Truffle – Create Project

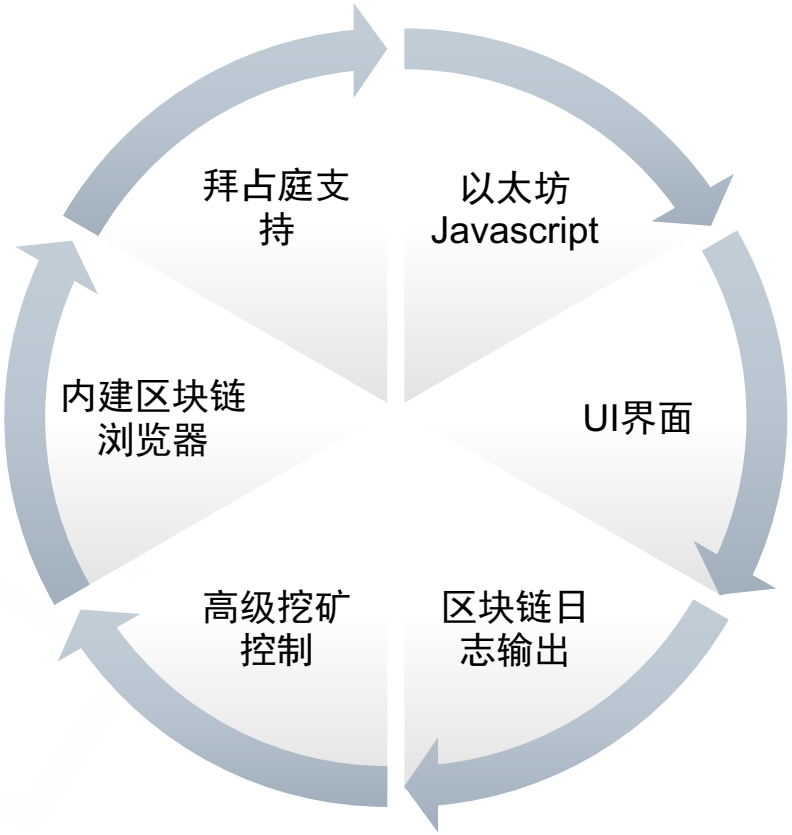
```
TsaiYee:demo Yee$ mkdir truffle-demo
TsaiYee:demo Yee$ cd truffle-demo/
TsaiYee:truffle-demo Yee$ truffle init
Downloading...
Unpacking...
Setting up...
Unbox successful. Sweet!

Commands:

  Compile:        truffle compile
  Migrate:        truffle migrate
  Test contracts: truffle test
TsaiYee:truffle-demo Yee$ ls
contracts       test            truffle.js
migrations      truffle-config.js
TsaiYee:truffle-demo Yee$
```

Truffle – Compile&Migrate&Test

```
TsaiYee:truffle-demo Yee$ truffle develop
Truffle Develop started at http://localhost:9545/

Accounts:
(0) 0x627306090abab3a6e1400e9345bc60c78a8bef57
(1) 0xf17f52151ebef6c7334fad080c5704d77216b732

Private Keys:
(0) c87509a1c067bbde78beb793e6fa76530b6382a4c0241e5e4a9ec0a0f44dc0d3
(1) ae6ae8e5ccbfb04590405997ee2d52d2b330726137b875053c36d94e974d162f

Mnemonic: candy maple cake sugar pudding cream honey rich smooth crumble sweet treat

truffle(develop)> compile
Compiling ./contracts/Migrations.sol...
Writing artifacts to ./build/contracts

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

Network up to date.
truffle(develop)>test
Using network 'develop'.


  0 passing (1ms)
```

开发框架 Ganache





目录

实战案例—撸猫

KittyCore.sol

```
pragma solidity ^0.4.11;
/**
 * @title Ownable
 * @dev The Ownable contract has an owner address, and provides basic authorization
 * functions, this simplifies the implementation of "user permissions".
 */
contract Ownable {
  address public owner;
  /**
   * @dev The Ownable constructor sets the original `owner` of the contract to the s
   * account.
   */
  function Ownable() {
    owner = msg.sender;
  }
  /**
   * @dev Throws if called by any account other than the owner.
   */
  modifier onlyOwner() {
    require(msg.sender == owner);
    _;
  }
  /**
   * @dev Allows the current owner to transfer control of the contract to a newOwner
   * @param newOwner The address to transfer ownership to.
   */
  function transferOwnership(address newOwner) onlyOwner {
    if (newOwner != address(0)) {
      owner = newOwner;
    }
  }

}
/// @title Interface for contracts conforming to ERC-721: Non-Fungible Tokens
/// @author Dieter Shirley <dete@axiomzen.co> (https://github.com/dete)
contract ERC721 {
    // Required methods
    function totalSupply() public view returns (uint256 total);
    function balanceOf(address _owner) public view returns (uint256 balance);
    function ownerOf(uint256 _tokenId) external view returns (address owner);
    function approve(address _to, uint256 _tokenId) external;
    function transfer(address _to, uint256 _tokenId) external;
    function transferFrom(address _from, address _to, uint256 _tokenId) external;

    // Events
    event Transfer(address from, address to, uint256 tokenId);
    event Approval(address owner, address approved, uint256 tokenId);

    // Optional
    // function name() public view returns (string name);
    // function symbol() public view returns (string symbol);
    // function tokensOfOwner(address _owner) external view returns (uint256[] token
    // function tokenMetadata(uint256 _tokenId, string _preferredTransport) public v

```

KittyCore.sol | CryptoKittiesSalesAuction.sol

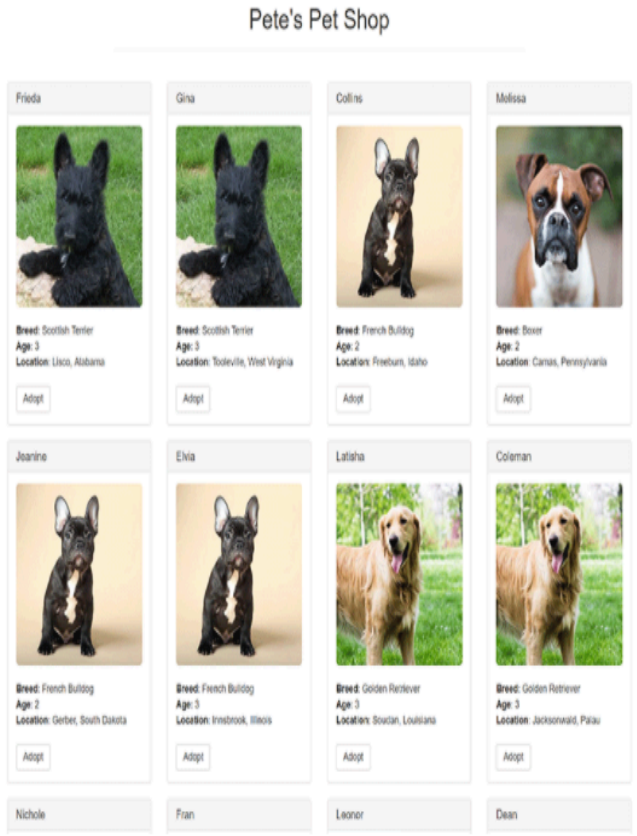
```
/// @title Auction Core
/// @dev Contains models, variables, and internal methods for the auction.
/// @notice We omit a fallback function to prevent accidental sends to this contract.
contract ClockAuctionBase {
    // Represents an auction on an NFT
    struct Auction {
        // Current owner of NFT
        address seller;
        // Price (in wei) at beginning of auction
        uint128 startingPrice;
        // Price (in wei) at end of auction
        uint128 endingPrice;
        // Duration (in seconds) of auction
        uint64 duration;
        // Time when auction started
        // NOTE: 0 if this auction has been concluded
        uint64 startedAt;
    }
    // Reference to contract tracking NFT ownership
    ERC721 public nonFungibleContract;
    // Cut owner takes on each auction, measured in basis points (1/100 of a percent).
    // Values 0-10,000 map to 0%-100%
    uint256 public ownerCut;
    // Map from token ID to their corresponding auction.
    mapping (uint256 => Auction) tokenIdToAuction;
    event AuctionCreated(uint256 tokenId, uint256 startingPrice, uint256 endingPrice, uint256
    event AuctionSuccessful(uint256 tokenId, uint256 totalPrice, address winner);
    event AuctionCancelled(uint256 tokenId);
    /// @dev Returns true if the claimant owns the token.
    /// @param _claimant - Address claiming to own the token.
    /// @param _tokenId - ID of token whose ownership to verify.
    function _owns(address _claimant, uint256 _tokenId) internal view returns (bool) {
        return (nonFungibleContract.ownerOf(_tokenId) == _claimant);
    }
    /// @dev Escrows the NFT, assigning ownership to this contract.
    /// Throws if the escrow fails.
    /// @param _owner - Current owner address of token to escrow.
    /// @param _tokenId - ID of token whose approval to verify.
    function _escrow(address _owner, uint256 _tokenId) internal {
        // it will throw if transfer fails
        nonFungibleContract.transferFrom(_owner, this, _tokenId);
    }
    /// @dev Transfers an NFT owned by this contract to another address.
    /// Returns true if the transfer succeeds.
    /// @param _receiver - Address to transfer NFT to.
    /// @param _tokenId - ID of token to transfer.
    function _transfer(address _receiver, uint256 _tokenId) internal {
        // it will throw if transfer fails
        nonFungibleContract.transfer(_receiver, _tokenId);
    }
    /// @dev Adds an auction to the list of open auctions. Also fires the
    ///  AuctionCreated event.
    /// @param _tokenId The ID of the token to be put on auction.
    /// @param _auction Auction to add.
```

实战案例—Pet Shop

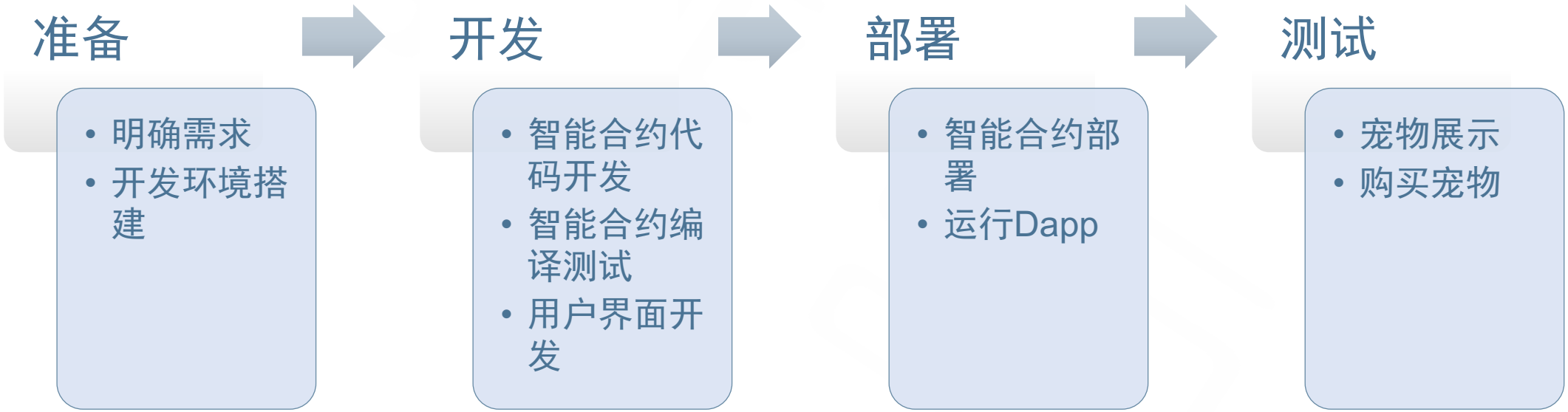
index.html | pets.json | app.js | Adoption.sol

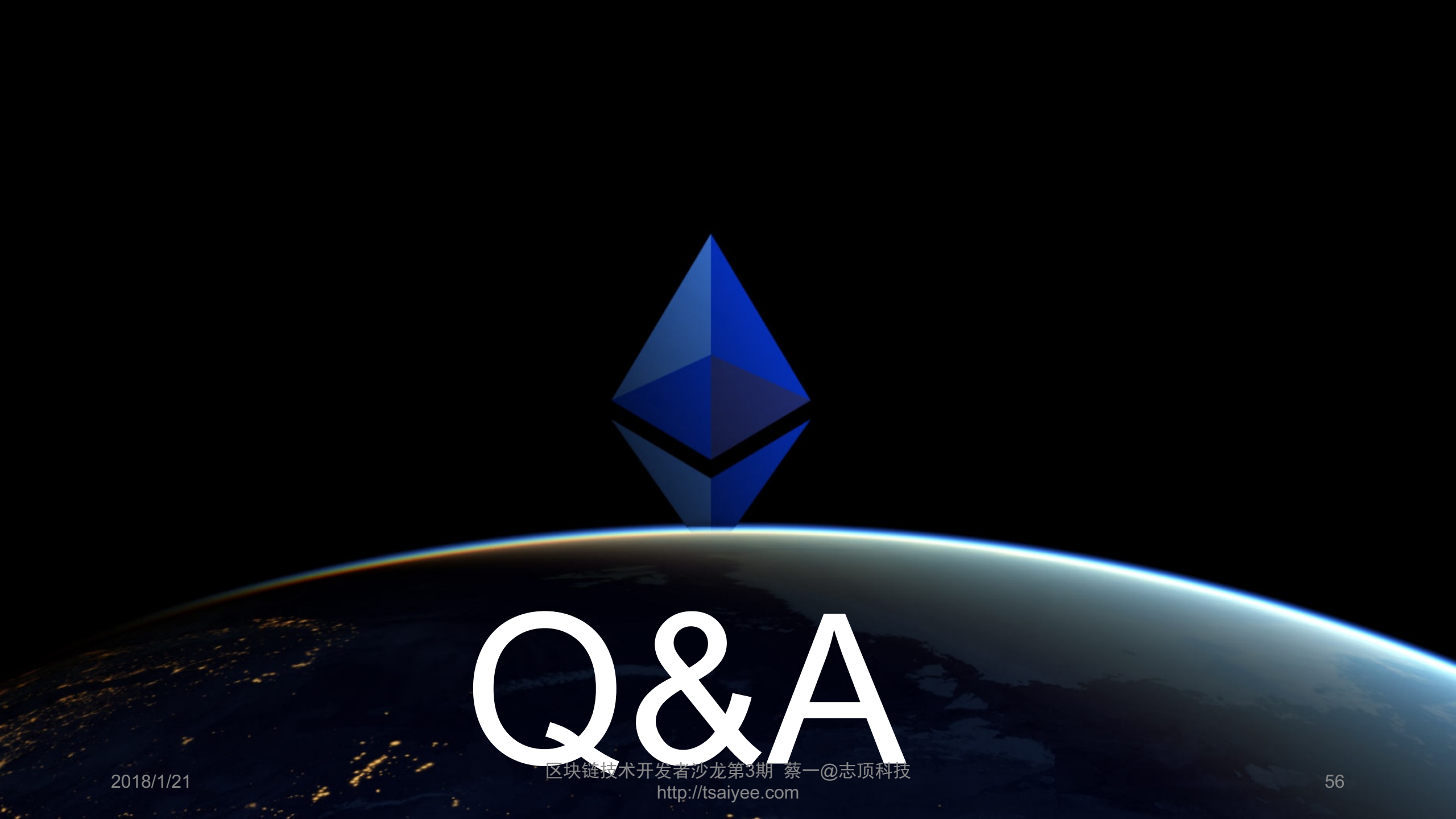
```
pragma solidity ^0.4.17;

contract Adoption {
    address[16] public adopters;

    // Adopting a pet
    function adopt(uint petId) public returns (uint) {
        require(petId >= 0 && petId <= 15);
        adopters[petId] = msg.sender;
        return petId;
    }
    // Retrieving the adopters
    function getAdopters() public view returns (address[16])
        return adopters;
    }
}
```

```
initWeb3: function() {
  // Is there an injected web3 instance?
  if (typeof web3 !== 'undefined') {
    App.web3Provider = web3.currentProvider;
  } else {
    // If no injected web3 instance is detected, fall back to
    App.web3Provider = new Web3.providers.HttpProvider('http:/
  }
  web3 = new Web3(App.web3Provider);

  return App.initContract();
},

initContract: function() {
  $.getJSON('Adoption.json', function(data) {
    // Get the necessary contract artifact file and instantiat
    var AdoptionArtifact = data;
    App.contracts.Adoption = TruffleContract(AdoptionArtifact)

    // Set the provider for our contract
    App.contracts.Adoption.setProvider(App.web3Provider);

    // Use our contract to retrieve and mark the adopted pets
    return App.markAdopted();
  });

  return App.bindEvents();
},
```



实战案例演示—Pet Shop




Q&A