

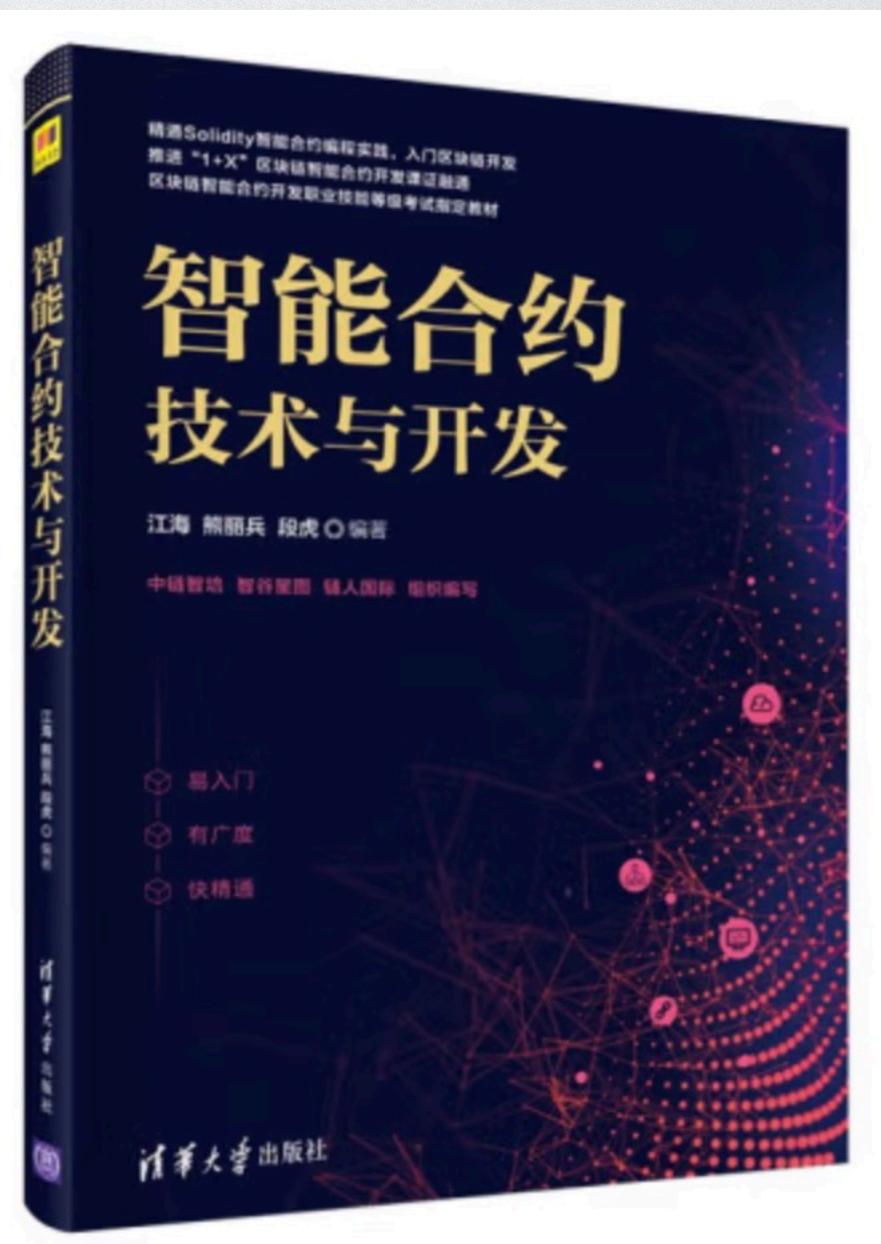
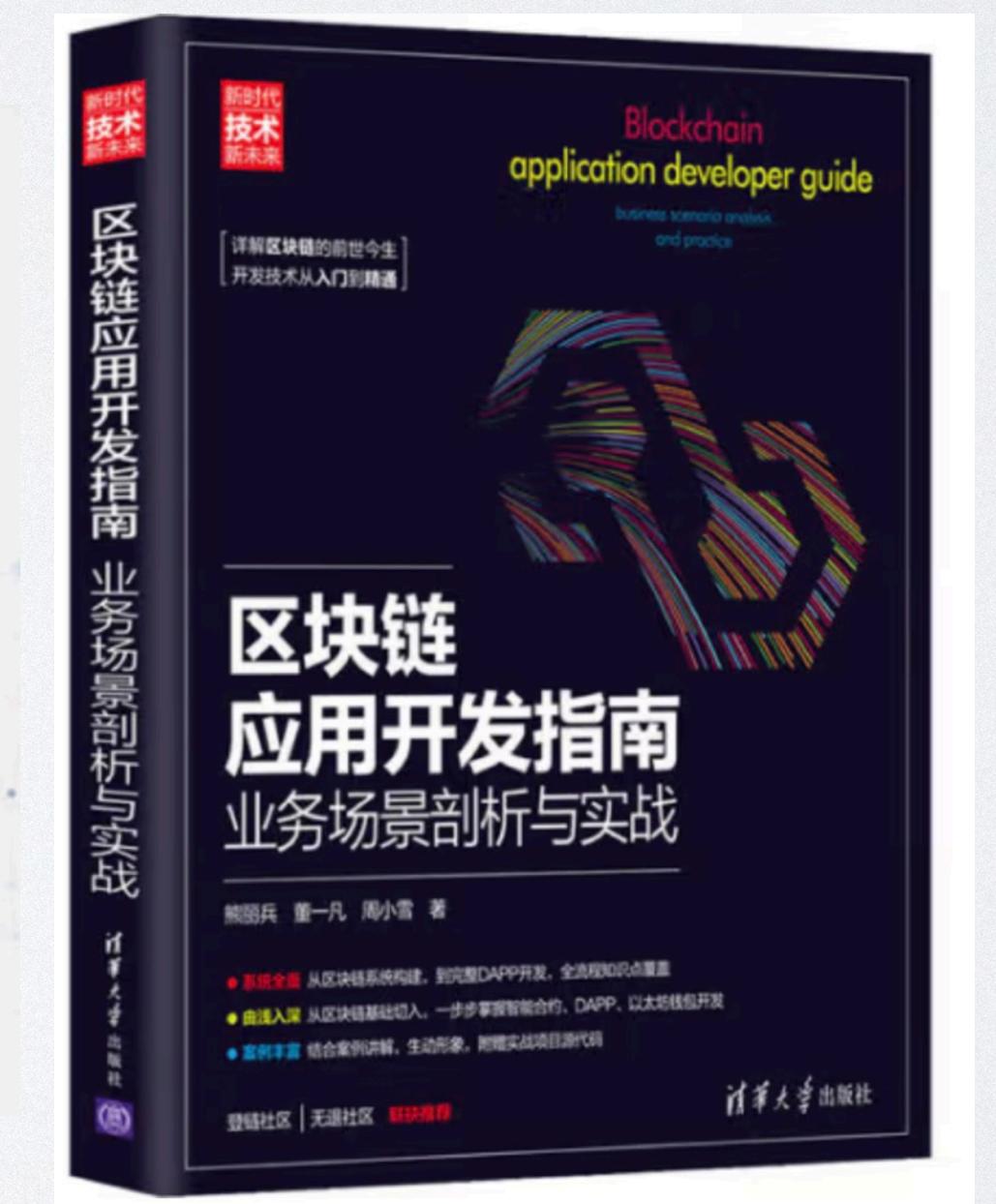
区块链集训营

二期

登链社区 - Tiny熊

ABOUT ME

- Tiny 熊
- 创新工场（点心）、猎豹移动、合伙创业
- 登链社区发起人 (from 2017)
- 出版过几本区块链书籍：



课程介绍

- 视频直播、实时讨论
- 代码开源：https://github.com/xilibi2003/training_camp_2
- 图文教程：<https://decert.me/tutorial/solidity/intro/>
- 课后录播巩固、习题练习
- 微信群答疑
- 优秀学员推荐就业

要求

- 较好的计算机背景
- GitHub
- 命令行
- 有至少一门语言编程经验

区块链及智能合约平台介绍

- 以太坊平台详解：智能合约平台、账户、GAS机制等
- 了解核心概念：账号及合约等、交易和消息调用、货币单位及Gas、钱包等
- 以一个简单的智能合约为例，介绍合约编译、部署、测试。
- 开发工具介绍如：Remix、Truffle、Ganache、Hardhat、Foundry等

SOLIDITY语言主要特性

- Solidity 基本类型、数组、结构体、映射
- Solidity API 介绍
- 合约函数、函数修改器、函数修饰符，及各类特殊函数
- 错误处理、合约继承、接口、库及 Openzeppelin 合约库
- 理解合约事件
- 理解ABI
- 合约代理、合约升级、Multicall

智能合约及WEB3开发基础

- 最常用 ERC 标准介绍及实战：ERC20、ERC777、EIP2612、ERC721、ERC1155
- Hardhat 实战技巧：自动化脚本、代码验证、导出 ABI 等。
- Foundry 开发框架及测试应用
- 优化 GAS 的众多技巧
- DApp 开发：前端与合约交互（ethers.js）

WEB3 应用开发必备

- wagmi 前端开发库
- 解析合约事件与TheGraph 使用
- NFTScan 相关 API
- Oracle 语言机、Keeper 服务等
- 抽象账户(AA)/智能钱包/多签钱包
- IPFS、Layer2、跨链等

DEX 分析专场

- Uniswap V2 AMM模型
- 无常损失与 AMM 滑点
- SushiSwap 如何抢Uni 的流动性及质押模型介绍
- Uniswap v3 的创新
- 相关 DEX 协议介绍：0x协议、Curve 协议

抵押借贷及衍生品协议

- Compound 借贷利率模型分析
- AAVE 闪电贷及应用
- DyDx 衍生品交易
- Perpetual vAmm 杠杆交易

稳定币及 DAO

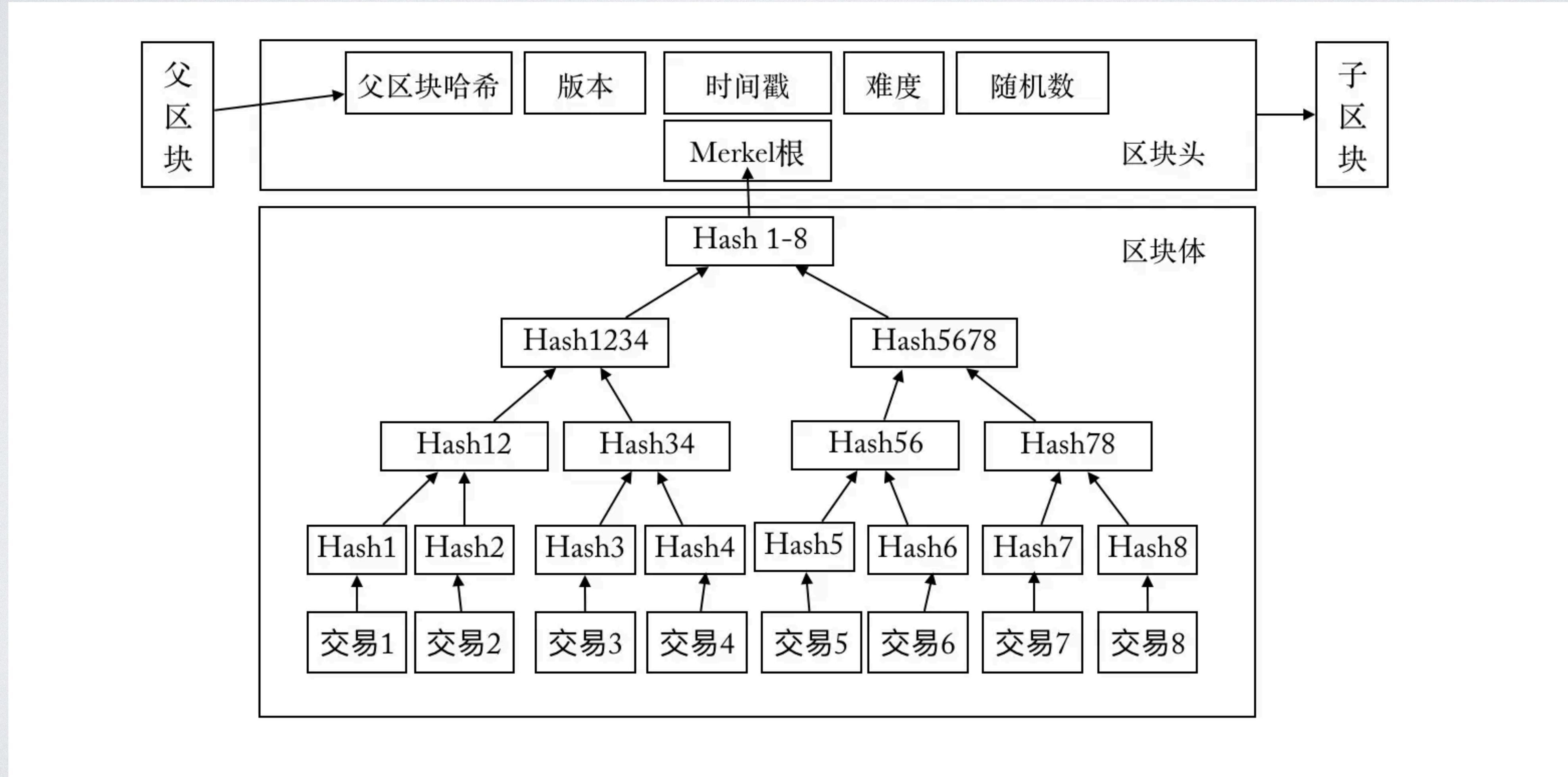
- MakerDAO 稳定币DAI
- 算法稳定币介绍：FRAX FEI OHM 等
- DAO 治理
- 案例分享及讨论

理解区块链及智能合约开发

什么是区块链

- 通过hash 串联的区块结构
- 区块内的交易通过 Merkel 树保存
- 一个由多个节点组成的网络（公链）

什么是区块链



区块链意义

- 解耦写代码、运行代码 -> 透明、信任
- 无法篡改的历史、无法抵赖 -> 可追溯

什么是以太坊

- 一台世界计算机（去中心化，任何人都可使用）
- 一个状态机（由交易触发的状态转换系统）
- 一个智能合约平台（计算平台）

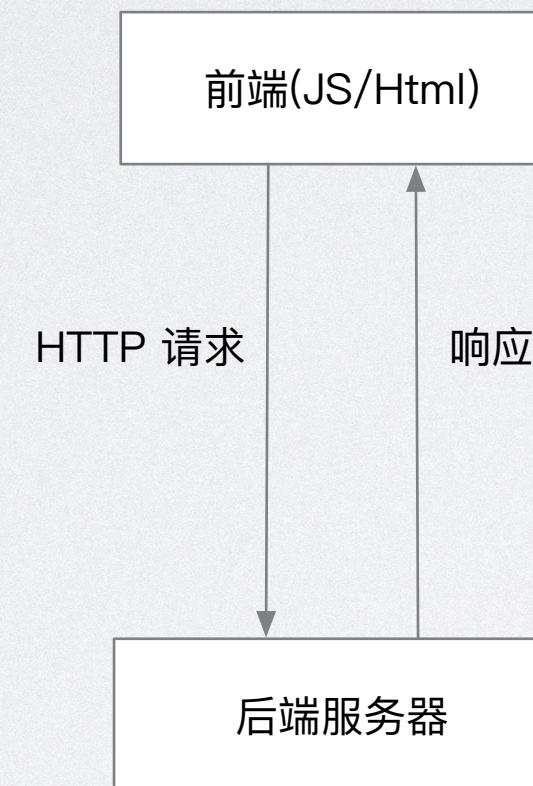
智能合约

- 智能：可执行（独立性、不受干扰）
- 合约：协议、规则

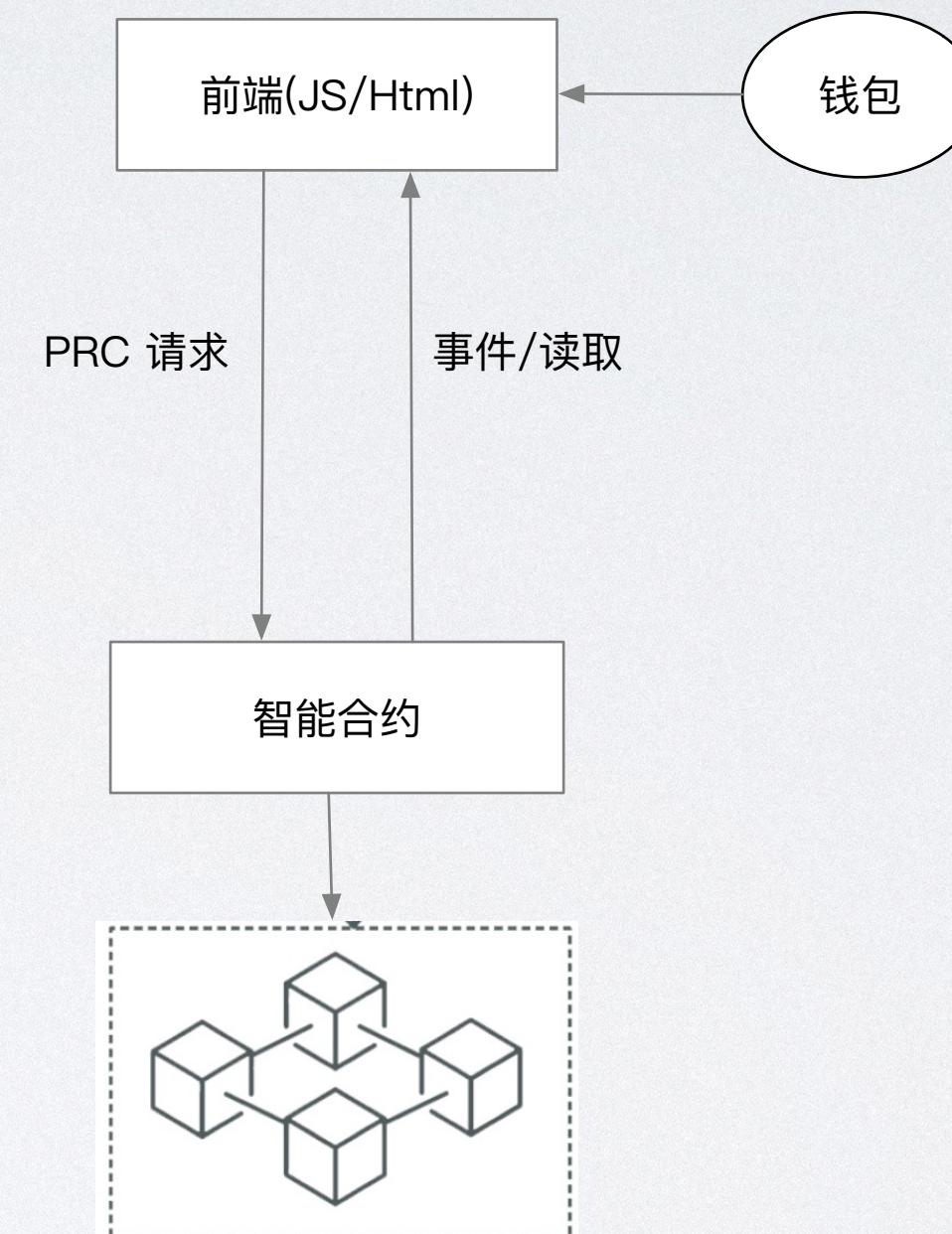
链上执行的程序，是代码和数据（状态）的集合

WEB3应用中的智能合约

传统应用



Web3 应用



核心逻辑使用智能合约运行在区块链的上，实现去信任。
用户数据通过钱包由用户自己管理。

智能合约

- Solidity(.sol): 智能合约开发语言

- 右边是一个简单的计数器
- Counter: 合约状态变量，保存在链上
- count(): 合约函数

```
pragma solidity ^0.8.0;
contract Counter {
    uint public counter;
    constructor() {
        counter = 0;
    }
    function count() public {
        counter = counter + 1;
    }
}
```

智能合约代码结构

- 编译器声明：pragma
- 合约声明: contract/ interface
- 状态变量（注明类型）, 类型定义
- 函数
- 事件
- 错误定义 (Error)

```
pragma solidity ^0.8.0;
contract Counter {
    uint public counter;
    constructor() {
        counter = 0;
    }
    function count() public {
        counter = counter + 1;
    }
}
```

智能合约

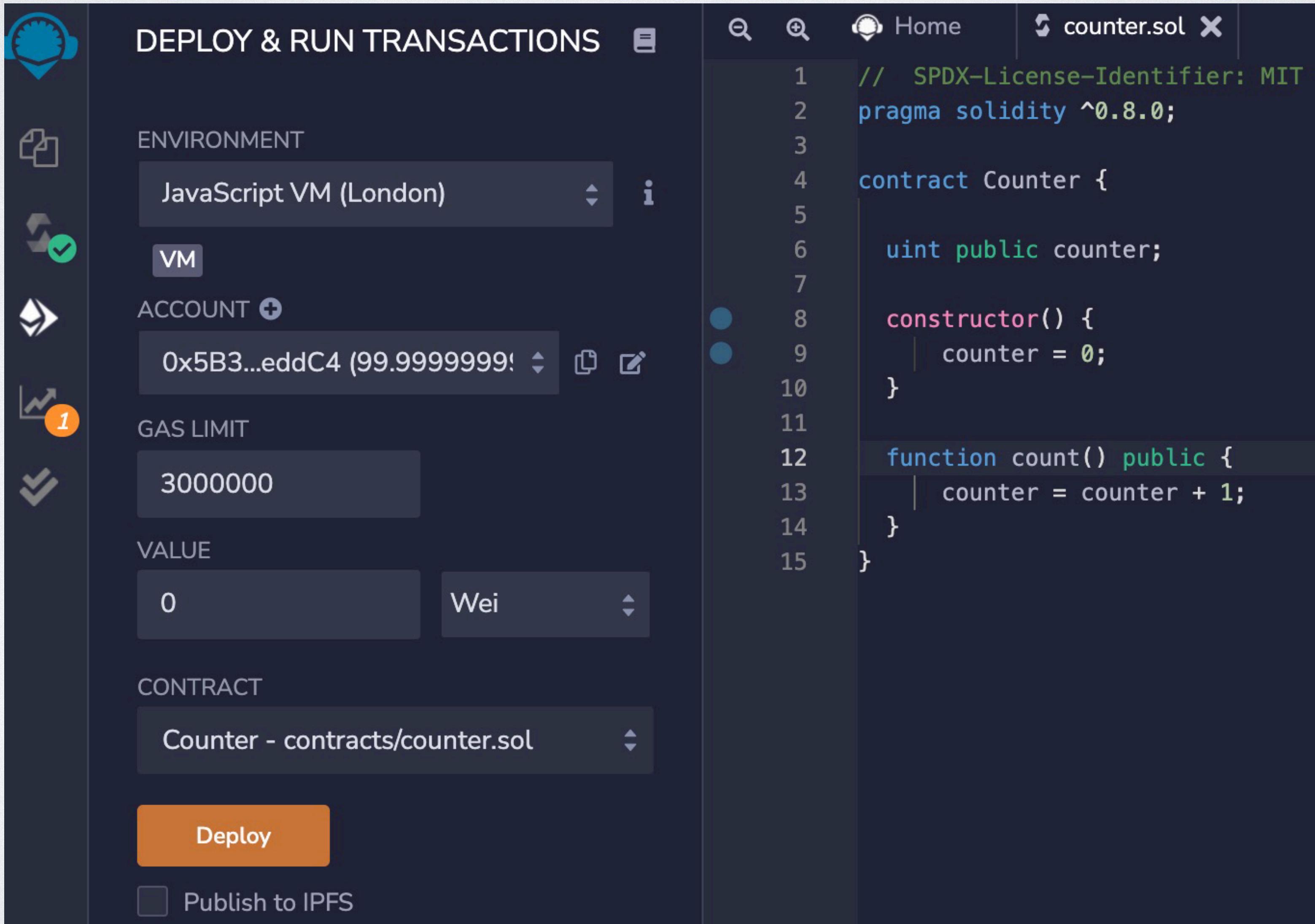
- Remix IDE: Solidity Online IDE

<https://remix.ethereum.org>

DEMO

REMIX

<https://remix.ethereum.org>



The screenshot shows the REMIX interface with the following configuration:

- ENVIRONMENT:** JavaScript VM (London)
- ACCOUNT:** 0x5B3...eddC4 (99.9999999)
- GAS LIMIT:** 3000000
- VALUE:** 0 Wei
- CONTRACT:** Counter - contracts/counter.sol

The code editor displays the Solidity code for the 'Counter' contract:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

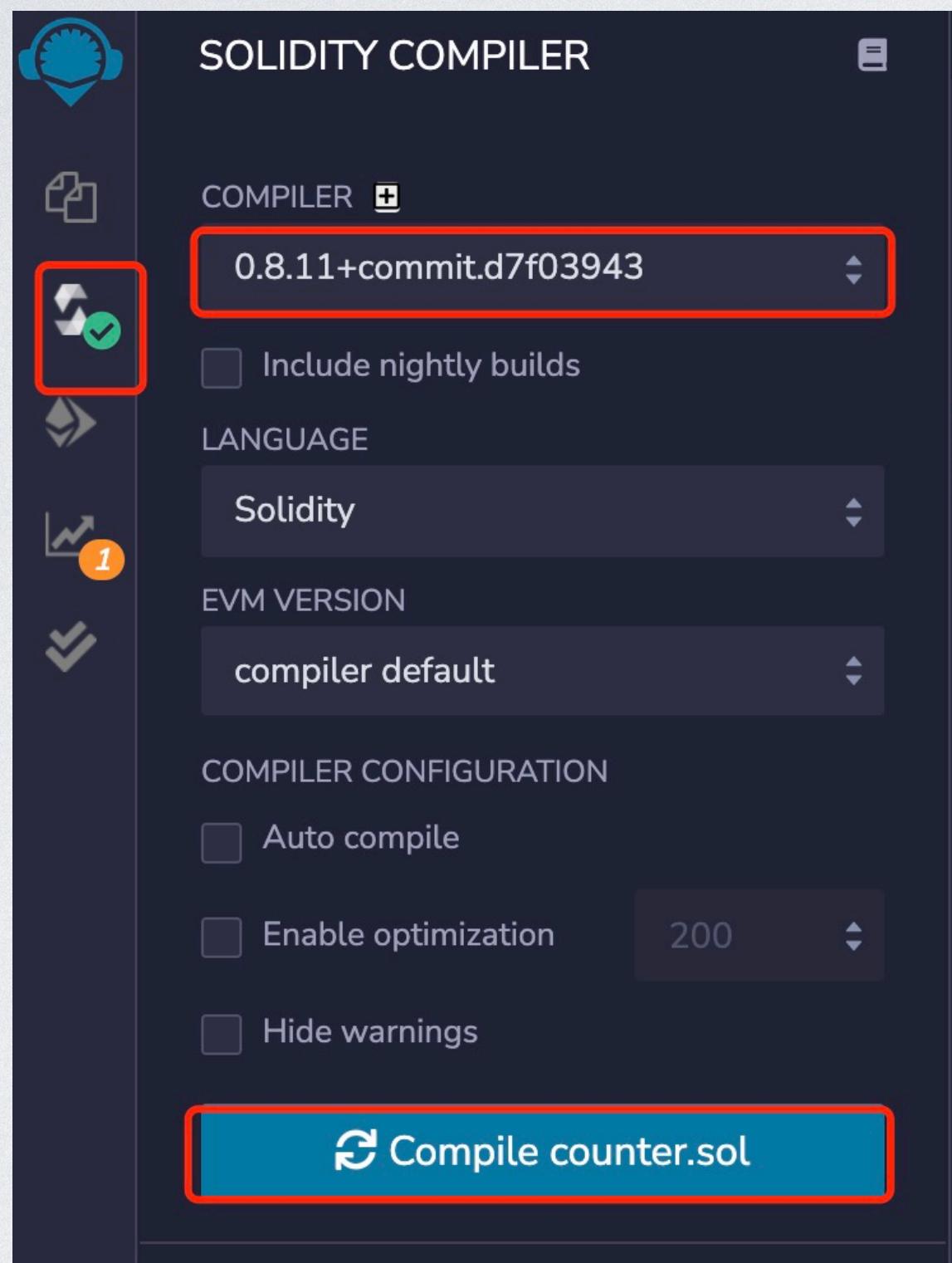
contract Counter {
    uint public counter;
    constructor() {
        counter = 0;
    }
    function count() public {
        counter = counter + 1;
    }
}
```

A large orange button at the bottom left is labeled "Deploy".

第1周

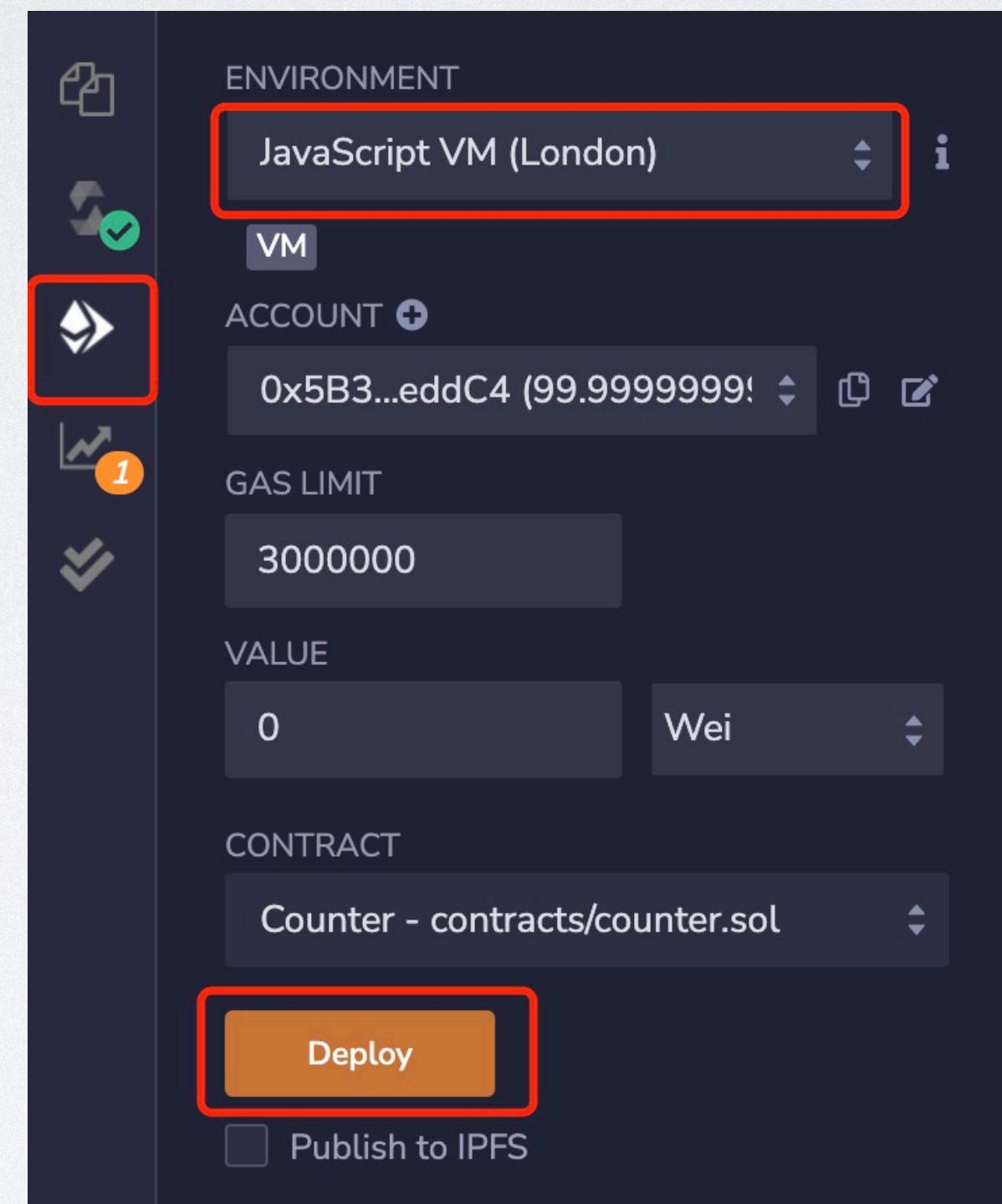
REMIX

<https://remix.ethereum.org>



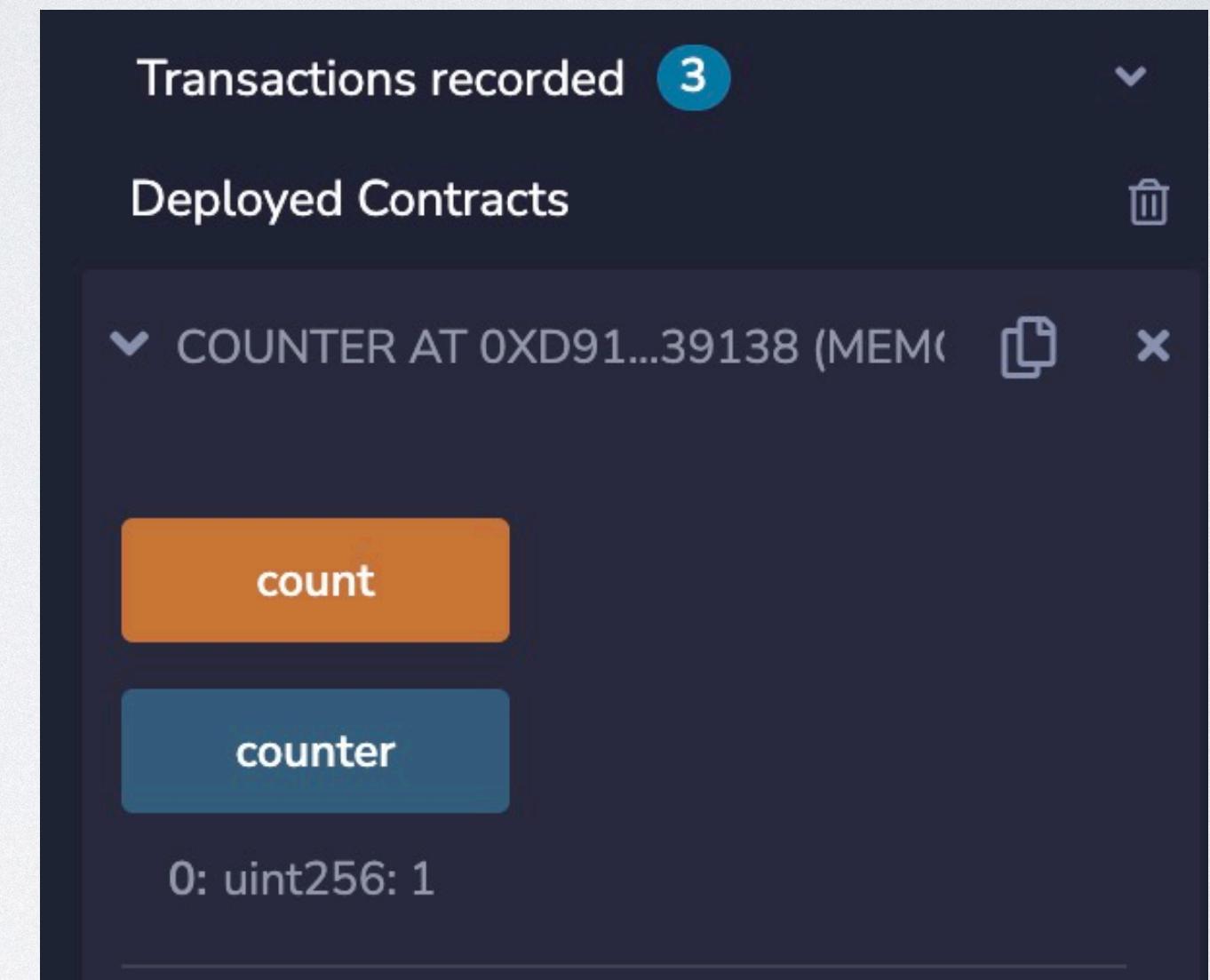
可选择编译器版本

编译 产出：ABI 与 字节码



可部署到不同的网络

部署 产出：合约地址

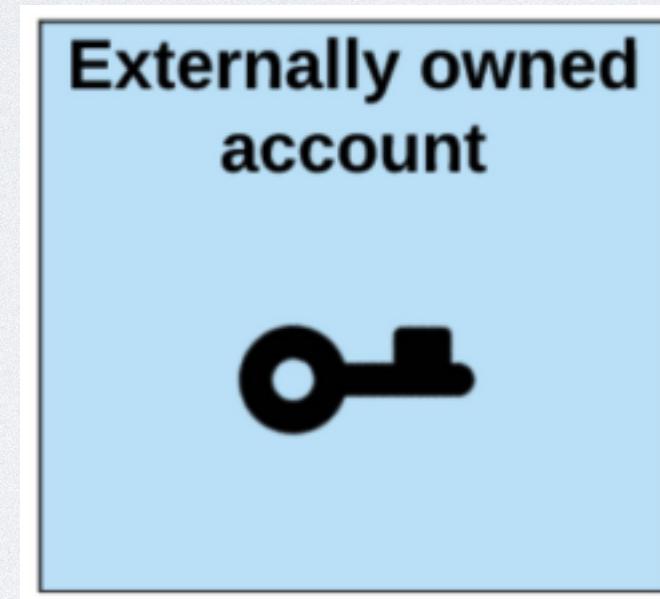


执行合约函数：
橙色：触发交易
蓝色：仅读取

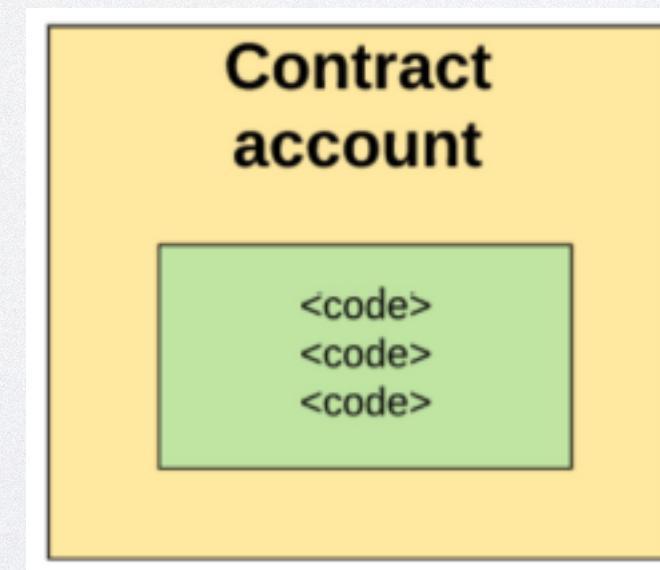
执行

账户

- 外部账户 (EOA)：由私钥控制，妥善保管、不可恢复



- 合约账户：代码控制



都用 20 个字节表示：
0xea674fdde714fd979...

账户

- 外部账户 (EOA) 与 合约账户在 EVM 层面是等效的，都是有：nonce (交易序号)、balance (余额)、storageRoot (状态)、codeHash (代码)

账户结构



Smart Contract

Ethereum Account Type (Just like User Account)



Address

0x16E0022b17B...

0 Ether



Balance

```
contract Counter {  
    uint counter;
```



Code

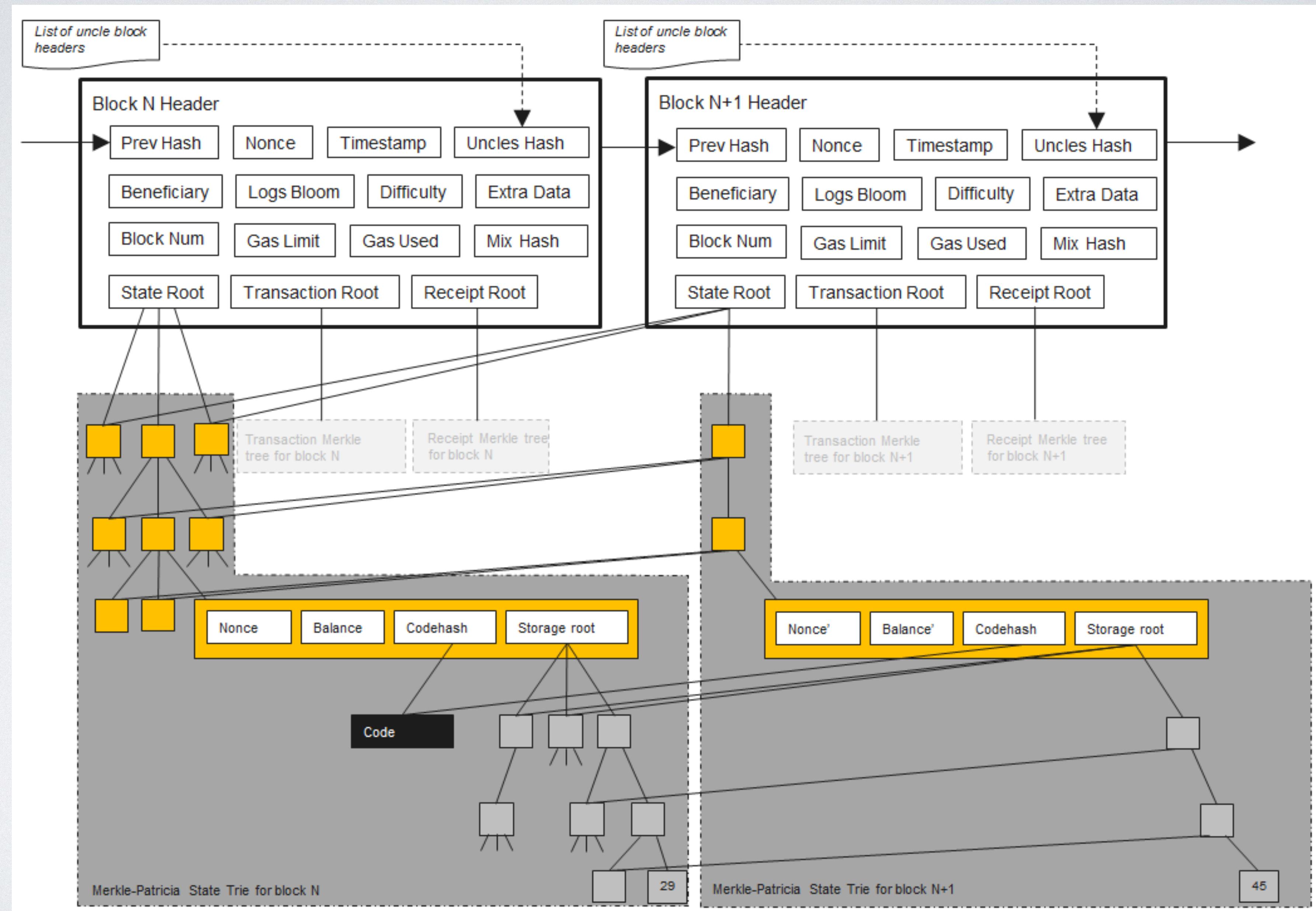
```
function Counter() public {  
    counter = 0;  
}
```



State

```
function count() public {  
    counter = counter + 1;  
}
```

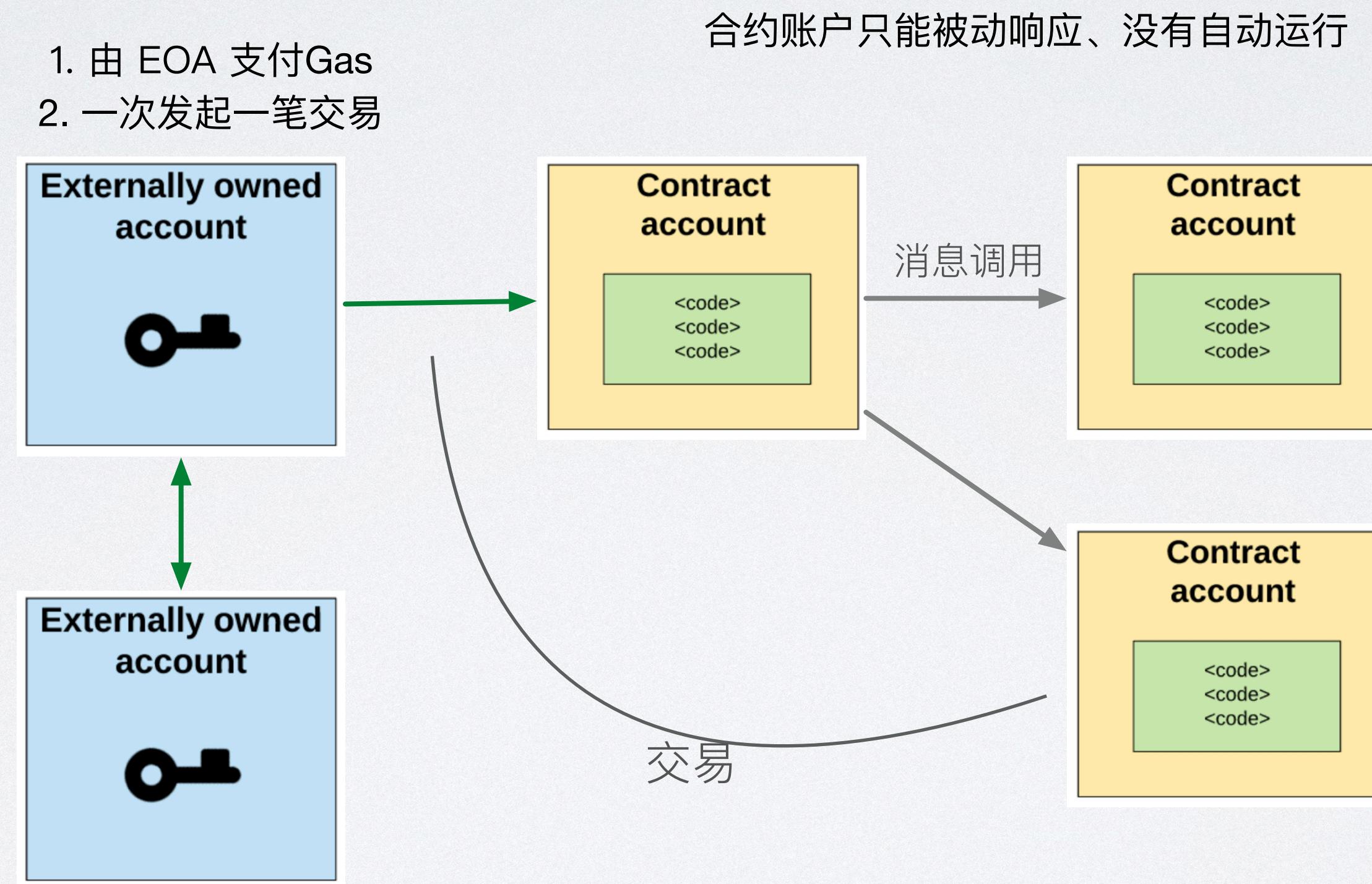
第1周



账户

- 但是在表现上有不一样：
 - **交易**只能从外部账号发出，合约只能被动相应执行。
 - 合约之间的交互通常称为**消息**，所有的手续费 gas 只能由外部账号支付。

账户交互



交易：原子性

以太坊 三种交易

- 普通交易
- 创建合约
- 调用合约

```
{  
  to: '0x687422...',  
  value: 0.0005,  
  data: "0x" // 也可以附加消息  
}
```

```
{  
  to: '',  
  value: 0.0,  
  data:"0x606060405234156100057x106....."  
}
```

```
{  
  to: '0x687422eEA2cB73..', //合约地址  
  value: 0.0,  
  data: "0x06661abd"  
}
```

EVM

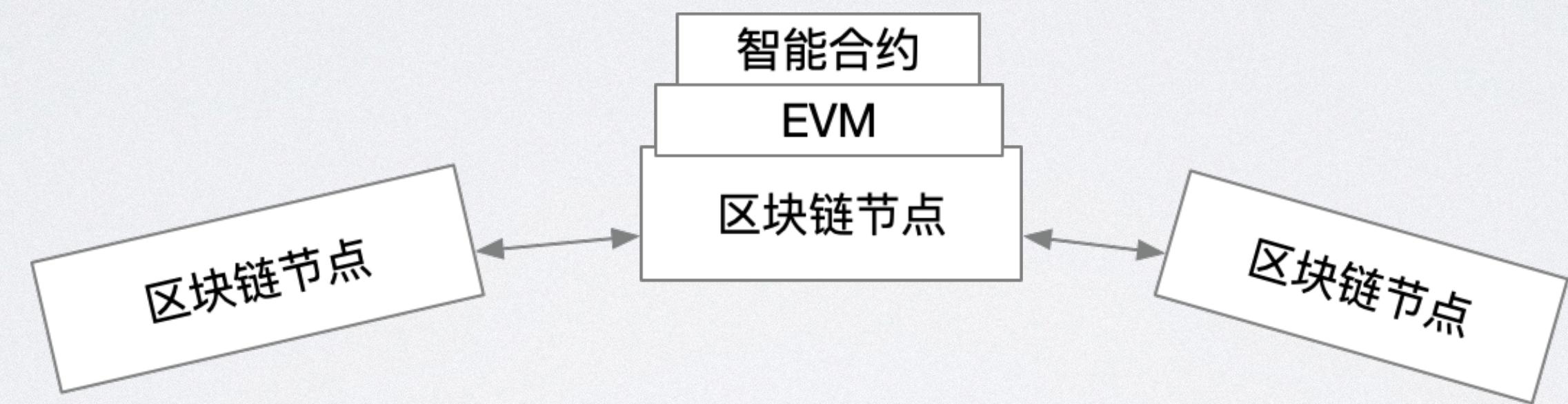
- EVM：以太坊虚拟机，智能合约执行环境
 - 类似 Java 至于 JVM
 - EVM 是一个封闭环境（不可访问外部系统）

以太坊强大的生态催生出来很多 EVM 兼容链: BSC , Polygon, OEC, Fantom ...

以太坊客户端

- 以太坊客户端：EVM 载体、网络中的节点程序
- 只要符合共识-规范，（几乎）任何语言都可以实现客户端
- 常见的客户端 (TheMerge 之后)
 - 执行层：Geth (Go 实现) 、 Nethermind(C#实现)、 Erigon (go)
 - 共识层：Prysm (Go) 、 Lighthouse (Rust 实现)
- 通过 RPC 提供服务
 - 节点服务商: Infura, alchemy

EVM 与 节点



钱包

- 账号管理工具，进行签名发起交易（管理助记词、私钥）

- 钱包：

- Metamask 外号：小狐狸（插件、App）
- imtoken
- TrustWallet



METAMASK

下载时仔细查看 URL，谨防钓鱼

GAS

- EVM 的计价规则，也防止图灵死机问题。
- GAS 是一个工作量单位，复杂度越大，所需 gas 越多。
- 手续费用 = gas 数量(gas limit) * gas price 单价 (以太币计价 gwei)

GAS 机制

gas (工作量单位)	汽油 (升)
gas Limit	提供多少升汽油 (用不完可退)
gas价格 (gwei)	每升汽油的价格 (元)
手续费： gas 用量 * gas价格	运费： 汽油用量 * 汽油价格
费用由发起交易的账号支付	谁提任务谁付运费

以太币单位

- 最小单位:Wei (伟)
- $10^9 \text{ Wei} = 1 \text{ Gwei}$
- $10^{12} \text{ Wei} = 1 \text{ szabo (萨博)}$
- $10^{15} \text{ Wei} = 1 \text{ finey (芬尼)}$
- $10^{18} \text{ Wei} = 1 \text{ Ether}$

区块链浏览器

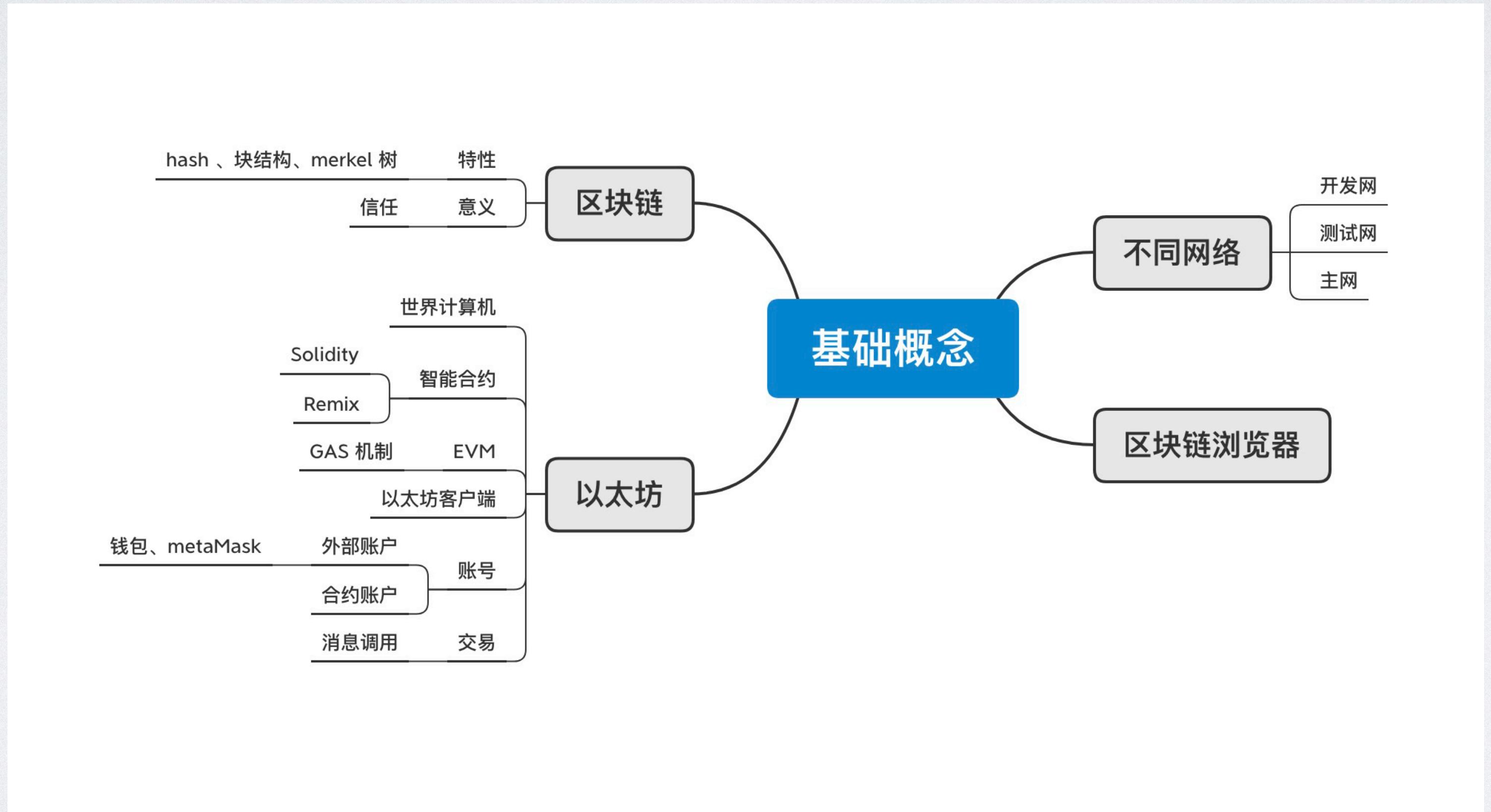
- 查看交易（交易hash、gas、）
- 查看Token信息
- 查看源代码

不同的网络

- 主网（价值网络） <https://cn.etherscan.com/>
- 测试网： <https://goerli.etherscan.io/>
- 开发模拟网（本地环境）

<https://chainlist.org/>

小结



Q & A

练习题

- 安装 Metamask、并创建好账号
- 执行一次转账
- 使用 Remix 创建一个Counter合约并部署:
 - Counter 合约有一个 add(x) 方法;

作业要求：

1. 使用自己的 github 创建一个作业代码库
2. 每一次作业使用一个文件夹(w1-1)
3. 提交代码、截图、交易 hash 等

习题解答

- 账号获取代币（水龙头或购买）
 - <https://goerlifaucet.com/>
- 使用 Remix 创建一个 Counter 合约并部署：
 - Counter 合约有一个 add(x) 方法；