TinTin

以太坊开发入门实战

owen liu from DODO

课程大纲

TinTin

第一节： 从与DApp应用交互开始，认识以太坊

第二节： **解刨合约交易，入门Solidity**

第三节： Solidity开发实战

第四节： 源码分析与合约安全

第五节： 链上数据记录与检索

第六节： 前端与合约的交互开发

第七节： 经典业务场景的合约解析

TinTin

解析上链交易的原始信息

```
let rawTransaction = {
    from: '0x7e83d9d94837ee82f0cc18a691da6f42f03f1d86',
    nonce: '0x152',
    gasPrice: '0xba43b7400',
    gasLimit: '0xc3500',
    value: '0x0',
    to: '0xba001E96AF87bF9d8D0BDA667067A9921FE6d294',
    data: '0xf87dc1b70000000000000000000000000eaa70c2a40820df9d38149c84dd943cfcb562587000000000000000000000000adcbae18580120667f7ff6c6451a426b13c67b70000000000000000
    chainId: 4
}
```

- from: 发起交易的账户地址
- nonce：账户下当前交易顺序值

可配合加速交易or取消交易

- gasPrice： gas对应的price
- gasLimit： gas上限
- chainId：网络编号

or EIP1559

普通转账

- to: 代币接收地址
- value：转账的数量
- data：'0x0'

创建合约

- to: 空
- value：'0x0'
- data：创建合约的code

与合约交互

- to: 合约地址
- value：'0x0' or 转入ETH数量
- data：调用合约的ABI 字段

TinTin

EIP1559 - 关于以太坊交易手续费机制的改进

Fees = 基础费（Base Fee） + 矿工小费（MinersTip）

- Base Fee：最终销毁，引入可变区块大小，即区块GasLimit 以目标1250万为标记，进行Base Fee的上下调整

- MinersTip：矿工小费

交易原始信息的参数调整：

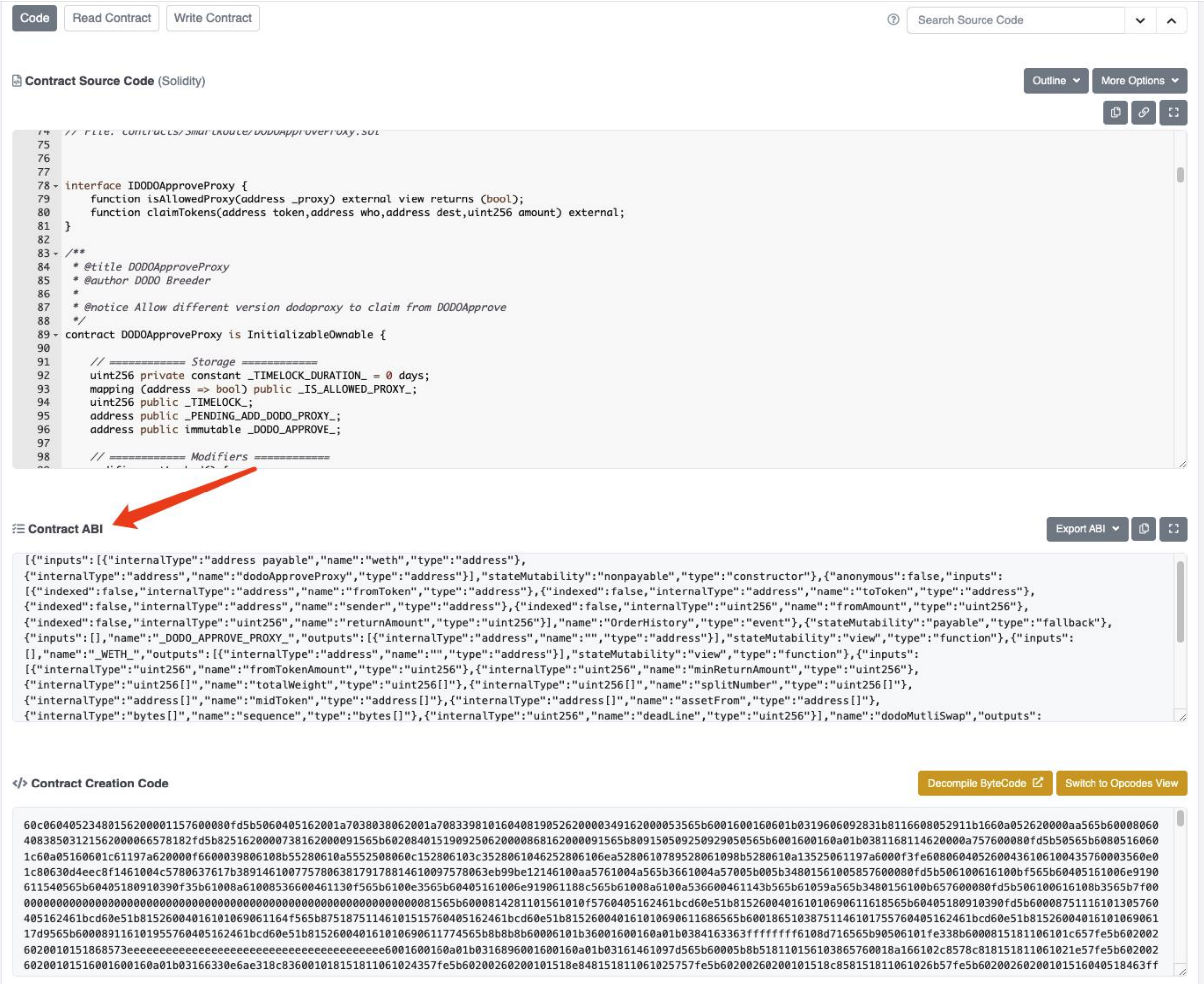
- from：发起交易的账户地址
- nonce：账户下当前交易顺序值
- gasPrice： gas对应的price
- gasLimit： gas上限
- chainId：网络编号

- **maxPriorityFeePerGas**: 最大每gas对应的矿工小费

- **maxFeePerGas**: 最大每gas对应的price，默认 2 * baseFeePerGas + maxPriorityFeePerGas, 并会退还未使用部分

ABI 编码：合约间调用或消息发送时的消息格式。来定义函数签名，参数编码，返回结果编码等。

JSON 定义



ABI 编码：对合约函数调用的data字段，包括针对函数签名以及调用参数进行ABI编码

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.16 <0.9.0;

contract Foo {
    function bar(bytes3[2] memory) public pure {}
    function baz(uint32 x, bool y) public pure returns (bool r) { r = x > 32 || y; }
    function sam(bytes memory, bool, uint[] memory) public pure {}
}
```

Thus for our `Foo` example if we wanted to call `baz` with the parameters `69` and `true`, we would pass 68 bytes total, which can be broken down into:

- `0xcdcd77c0`: the Method ID. This is derived as the first 4 bytes of the Keccak hash of the ASCII form of the signature `baz(uint32,bool)`.
- `0x0000000000000000000000000000000000000000000000000000000000000045`: the first parameter, a uint32 value `69` padded to 32 bytes
- `0x0000000000000000000000000000000000000000000000000000000000000001`: the second parameter - boolean `true`, padded to 32 bytes

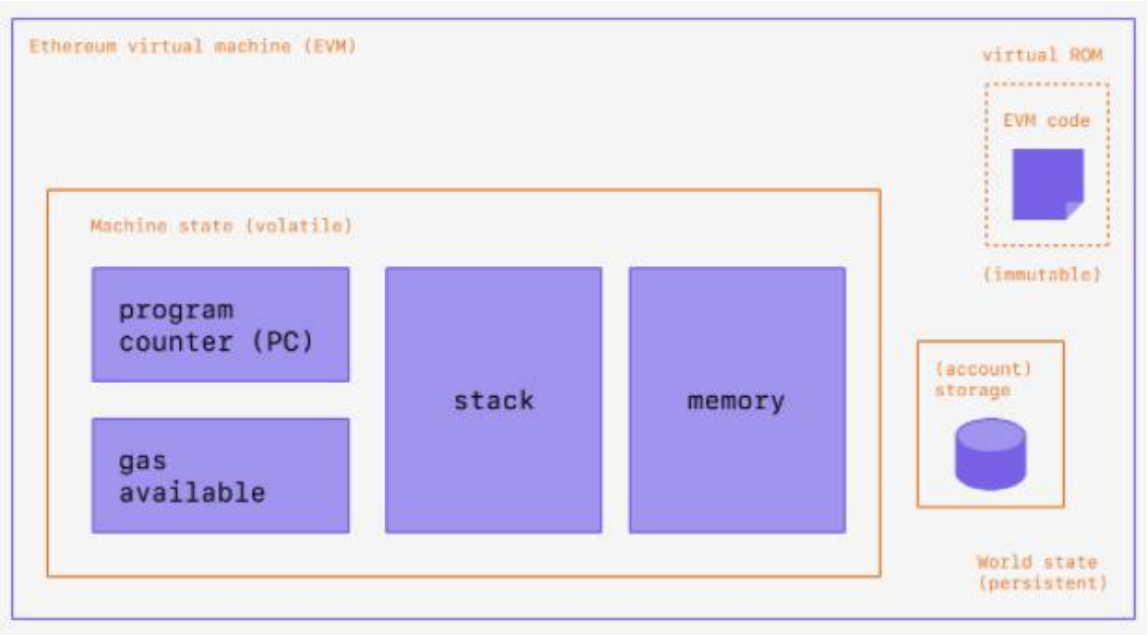
第二节：解刨合约交易，入门Solidity

TinTin

合约执行引擎：EVM

EVM 定义了从一个区块计算产生下一个区块链上有效状态的规则

以太坊作为一种分布式状态机。 其状态是一个大型数据结构，它不仅保存所有帐户和余额，而且还保存一个机器状态，它可以根据预定义的一组规则在不同的区块之间进行更改，并且可以执行任意的机器代码。 在区块中更改状态的具体规则由 EVM 定义。

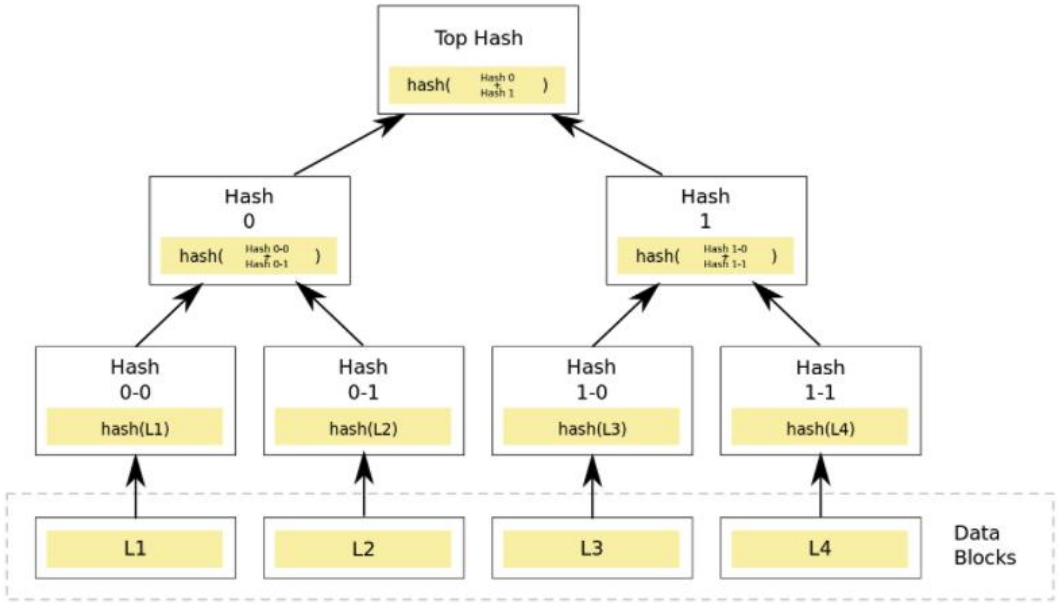


- 交易触发EVM执行，使得状态改变
- 状态是一个巨大的数据结构，称为 Merkle Patricia Tree
- 在给定输入的情况下，会产生确定性的输出

TinTin

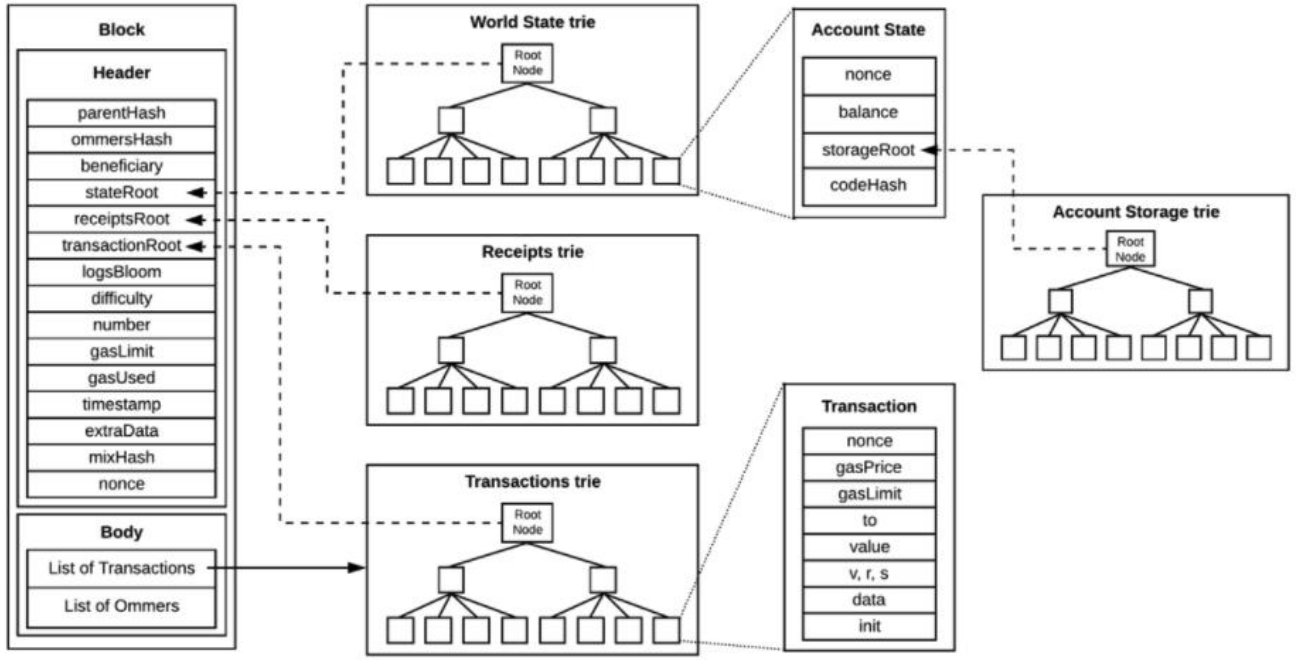
链上状态的数据结构

Merkle Tree



Merkle Patricia Tree

区块头包括三个树root值：交易树根，收据树根，状态树根



状态树包含了一个键值映射，其中键key是地址（账户地址&合约地址），而值value包括nonce,balance,codeHash以及storageRoot（storageRoot是Merkle树根，存储合约中的storage数据

TinTin

初识合约语言 Solidity

```sol
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.16 <0.9.0;

contract SimpleStorage {
    //状态变量
    address public owner;
    uint storedData;

    struct SimpleInfo { // 结构类型
        uint amount;
        address account;
    }

    enum SimpleState { Created, Locked, Inactive } // 枚举类型

    constructor() public {
        owner = msg.sender;
    }

    modifier onlyOwner() { //函数修饰器
        require(msg.sender == owner,"Only owner can call this.");
        _;
    }

    event Update(uint value); // 事件

    function set(uint x) public onlyOwner { //函数
        storedData = x;
        emit Update(x);
    }

    function get() public view returns (uint) { //函数
        return storedData;
    }
}
```

在 Solidity 中，合约类似于面向对象编程语言中的类，合约可以从其他合约继承。 每个合约中可以包含：

- 状态变量
- 函数
- 函数修饰器
- 事件
- 结构类型
- 枚举类型

课后扩展阅读：

- 关于EIP1559: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1559.md
- 关于ABI: https://docs.soliditylang.org/en/develop/abi-spec.html
- 关于状态数据结构：https://blog.ethereum.org/2015/11/15/merkling-in-ethereum/

TinTin

第二节课作业：

- 在测试网发送交易，还原原始交易信息字段
- 在测试网任意发送合约交易，解析data，获取调用合约的参数
- 阅读扩展文章

TinTin

TinTin小助手

TinTin公众号

Twitter | YouTube | Discord