以太坊学习：第六天

Solidity简介

- Solidity是一门面向合约的、为实现智能合约而创建的高级编程语言。这门语言收到了C++、Python和JavaScript语言的影响，设计的目的是能在行
- 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
- 内含的类型除了常见编程语言中的标准类型，还包括address等以太坊独有的类型，solidity源码文件通常以.sol作为扩展名
- 目前尝试solidity编程的最好方式是使用remix。remix是一个基于web浏览器的IDE，它可以让你编写solidity智能合约，然后部署并运行该智能合

Solidity语言特性

Solidity的语法接近于JavaScript，是一种面向对象的语言。但作为一种真正意义上运行在网络上的去中心化合约，它又有很多不同

- 以太坊底层基于账户，而不是UTXO，所以增加了一个特殊的address的数据类型用于定位用户和合约账户
- 语言内嵌框架支持支付。提供了payable等关键字，可以在语言基础层面直接支持支付
- 使用区块链进行数据存储。数据的每一个状态都可以永久保存，所以在使用时需要确定变量使用内存，还是区块链存储
- 运行环境是在去中心化的网络上，所以需要强调合约或函数执行的调用方式
- 不同异常机制。一旦出现异常，所有的执行都会被回撤，这主要是为了保证合约执行的原子性，以避免中间状态出现的数据不一致

Solidity源码和智能合约

- Solidity源代码要成为可以运行在以太坊上的智能合约需要经历如下步骤

1. 用Solidity编写的智能合约源代码需要先使用编译器编译为字节码（Bytecode），编写过程中会同时产生智能合约的二进制接口规范（Applicat Interface，简称ABI）；
2. 通过交易的方式将字节码部署到以太坊网络，每次成功部署都会产生一个新的智能合约账户
3. 使用JavaScript编写的DApp通常通过web3.js + ABI去调用智能合约中的函数来实现数据的读写

Solidity编译器

Remix：是一个基于Web浏览器的Solidity IDE，可在线使用而无需安装任何东西

solcjs：是Solidity源码库的构建目标之一，它是Solidity的命令行编译器，使用npm可以便捷地安装Solidity编译器solcjs。npm install -g solc

LEVI_104 0 0 0

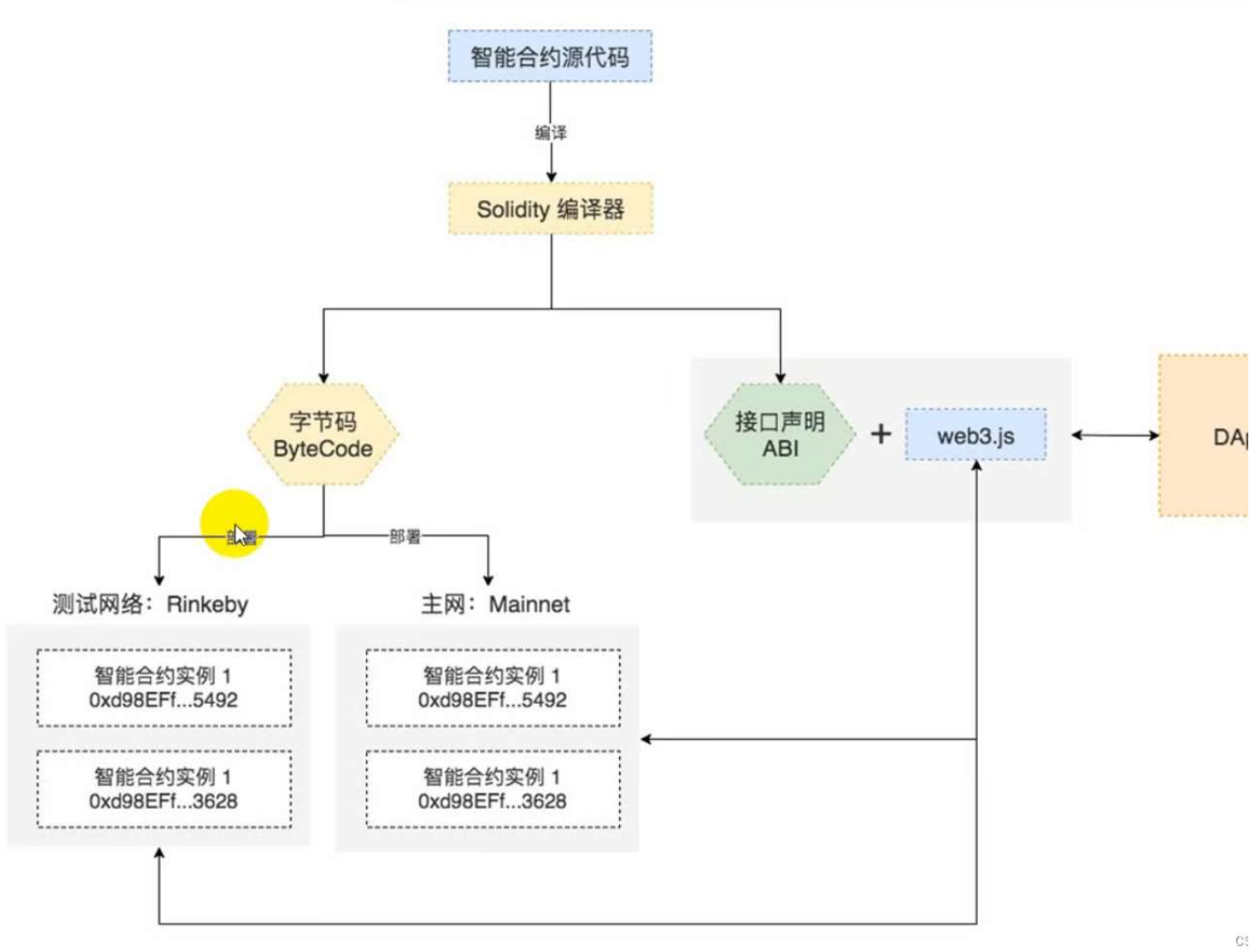

图1-1：尚硅谷区块链全套教程

简单投票合约

```
pragma solidity >=0.7.0 <0.9.0;

contract Ballot {

    struct Voter {
        uint weight;//权重：可以投票权重为1，不可以投票权重为0
        bool voted;//是否投过票
        address delegate;//有无代理人
        uint vote;//每一个提案用数字表示
    }

    struct Proposal {
        bytes32 name;//提案的名字
        uint voteCount;//这个提案的票数
    }

    address public chairperson;//投票项目的主席

    mapping(address => Voter) public voters;//一个地址对应一个投票者

    Proposal[] public proposals;//用于存放提案

    constructor(bytes32[] memory proposalNames) {
        chairperson = msg.sender;//我是主席
        voters[chairperson].weight = 1;//我的权重置为1：可以投票
        //初始化每一个提案：各个提案的名字为传入的名字数组。提案初始票数为0
        for (uint i = 0; i < proposalNames.length; i++) {
            proposals.push(Proposa
                name: proposalName
```

```
                voteCount: 031            }));
        }
    }

    //主席有权利：给其他人投票的权利
    function giveRightToVote(address voter) public {
        require(
            msg.sender == chairperson,
            "Only chairperson can give right to vote."
        );//只有主席才有权力
        require(
            !voters[voter].voted,
            "The voter already voted."
        );//投票人没有投票过才行
        require(voters[voter].weight == 0);//投票人要有投票的本身没有投票的权力
        voters[voter].weight = 1;
    }

    //找代理
    function delegate(address to) public {
        Voter storage sender = voters[msg.sender];//函数的调用者sender
        require(!sender.voted, "You already voted.");
        require(to != msg.sender, "Self-delegation is disallowed.");//不可以

        while (voters[to].delegate != address(0)) {//不断找到最终的被委托人（可以不断委托给其他人）
            //voters[to].delegate != address(0)：不能委托给零地址
            to = voters[to].delegate;//下一个被委托人
            //我委托给A，而A不可以再委托给我
            require(to != msg.sender, "Found loop in delegation.");
        }
        sender.voted = true;//委托之后，就相当于自己投过票了
        sender.delegate = to;//被委托人为to
        Voter storage delegate_ = voters[to];//被委托人to
        if (delegate_.voted) {//如果投过票了，所以就不可以再委托别人，只可以去投票了
            proposals[delegate_.vote].voteCount += sender.weight;//给提案投票
        } else {//如果没投过票
            delegate_.weight += sender.weight;//如果没投过票，那么自己的票权重增加
        }
    }

    //投票
    function vote(uint proposal) public {
        Voter storage sender = voters[msg.sender];//调用方法的人sender
        require(sender.weight != 0, "Has no right to vote");//没资格投票
        require(!sender.voted, "Already voted.");//已经投过票了
        require(proposal <= proposals.length,"out of the range of the array")//投票的范围
        sender.voted = true;
        sender.vote = proposal;
        proposals[proposal].voteCount += sender.weight;//投票
    }

    //返回获胜提案
    function winningProposal() public view returns (uint winningProposal_){
        uint winningVoteCount = 0;
        for (uint i = 0; i < proposals.length; i++) {
            if (proposals[i].voteCount > winningVoteCount) {
                winningVoteCount = proposals[i].voteCount;
                winningProposal_ = i;
            }
        }
    }

    //返回获胜提案的名字
    function winnerName() public view returns (bytes32 winnerName_){
        winnerName_ = proposals[winningProposal()].name;
    }
}
```

LEVI_104 0 0 0

“相关推荐”对你有帮助么？

非常没帮助 没帮助 一般 有帮助 非常有帮助

关于我们 招贤纳士 商务合作 寻求报道 400-660-0108 kefu@csdn.net 在线客服 工作时间 8:30-22:00

公安备案号11010502030143 京ICP备19004658号 京网文〔2020〕1039-165号 经营性网站备案信息 北京互联网违法和不良信息举报中心
家长监护 网络110报警服务 中国互联网举报中心 Chrome商店下载 ©1999-2022北京创新乐知网络技术有限公司 版权与免责声明 版权申诉
出版物许可证 营业执照

LEVI_104

0 0 0