قراردادهای هوشمند

Meetchain

Arshia Hemmat

01

مفاهیم و کلیات
قراردادهای هوشمند

02

چجوری برنامه‌نویس
بلاکچین بشیم

03

پیاده‌سازی قرارداد
هوشمند

01

مفاهیم و کلیات
قراردادهای هوشمند

تئوری

02

چجوری برنامه‌نویس
بلاکچین بشیم

تئوری

03

پیاده‌سازی قرارداد
هوشمند

عملی

01

مفاهیم و کلیات
قراردادهای هوشمند

تئوری

1 مشکلات قراردادهای سنتی

2 مزایای قراردادهای هوشمند

3 کاربرد قراردادهای هوشمند

02

چگونه برنامه‌نویس بلاکچین شویم

تئوری

1. انتخاب شبکه مناسب
2. بررسی زبان‌های موجود
3. انتخاب دوره آموزشی مناسب

03

پیاده‌سازی قرارداد هوشمند

عملی

1. معرفی محیط پیاده‌سازی
2. طراحی کانسپت‌های اولیه
3. شروع پیاده‌سازی قرارداد هوشمند

01 مقدمات و کلیات

مزایا و کاربردهای قراردادهای هوشمند

مشکلات قراردادهای سنتی

امکان نابودی و از بین رفتن قراردادها وجود دارد

امکان دستکاری در اطلاعات قرارداد برای طرفین قرارداد وجود دارد

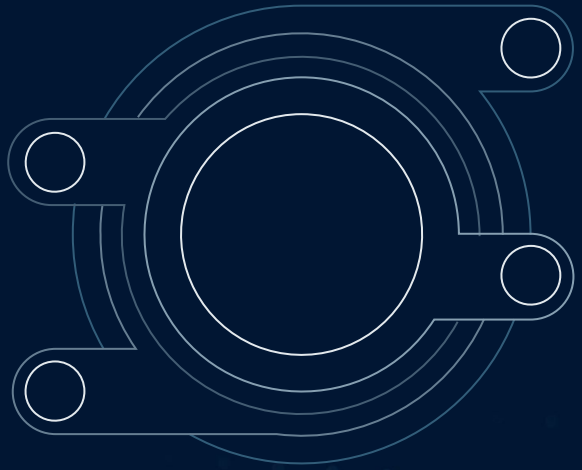
1

به شکل مکتوب برروی کاغذ

2

اشخاص ثالثی مانند وکیل یا دفتر خانه باید در فرایند دخالت کنند

مزایای قراردادهای هوشمند



امنیت غیرقابل تصور

صرفه جویی در زمان

کاهش خطاهای انسانی

کاهش هزینه‌ها

کاربرد قراردادهای هوشمند

سرمایه گذاری جمعی

معامله اموال اعتباری

طراحی و ساخت Dapp

02

برنامه‌نویسی بلاکچینی

بررسی شبکه‌های مختلف و زبان‌های مختلف

بلاکچینی بررسی پلتفرم‌های

- Tezos
- Stellar
- Hyperledger Fabric
- Hyperledger Sawtooth
- EOS
- Openchain
- Corda
- Tron
- Hedera Hashpraph
- Ethereum

منبع: https://www.leewayhertz.com/top-blockchain-platforms/

Ethereum

They have the largest community with core protocol developers, cypherpunks, crypto-economic researchers and mining organizations. It aims to eliminate internet third-parties who save data and track financial instruments.

features of the Ethereum platform:

- Smart Contracts Functionality
- Turing Completeness
- Permissioning
- Privacy
- Rapid Deployment
- Tokenization

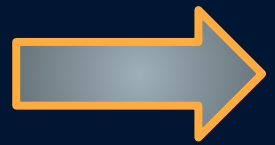
منبع: https://www.leewayhertz.com/top-blockchain-platforms/

Ethereum

They have the largest community with core protocol developers, cypherpunks, crypto-economic researchers and mining organizations. It aims to eliminate internet third-parties who save data and track financial instruments.

features of the Ethereum platform:

- Smart Contracts Functionality
- Turing Completeness
- Permissioning
- Privacy
- Rapid Deployment
- Tokenization



Solidity

03

پیاده‌سازی

بررسی محیط پیاده‌سازی و طراحی Dapp

محیط‌های پیاده‌سازی قراردادهای هوشمند

Remix

VS code

طراحی پروژه و یک داستان ساده!

طراحی پروژه و یک داستان ساده!

طراحی پروژه و یک داستان ساده!

طراحی پروژه و یک داستان ساده!

از همراهی شما سپاس‌گزاریم

@meetchain
zil.ink/meetchain

```
pragma solidity ^0.4.17;

contract lottary
{
    address public manager;
    address public winner_address;
    address[] public players;
    function lottary() public{
        manager = msg.sender;
    }
    function enter() public payable{
        require(msg.value > 0.01 ether);
        players.push(msg.sender);
    }
    function PickWinner() public CheckAdmin{
        uint index = random_generator() % players.length;
        players[index].transfer(this.balance);
        winner_address = players[index];
        players = new address[](0);
    }
    function random_generator() public view returns(uint){
        return uint(keccak256(block.difficulty, now, players));
    }
    function getPlayer() public view returns(address[]){
        return players;
    }
    function WinPrice() public view returns(uint)
    {
        return this.balance;
    }
    modifier CheckAdmin{
        require(msg.sender == manager);
        _;
    }

}
```

```
pragma solidity ^0.4.17;

contract lottary
{
    address public manager;
    address public winner_address;
    address[] public players;
    function lottary() public{
        manager = msg.sender;
    }
    function enter() public payable{
        require(msg.value > 0.01 ether);
        players.push(msg.sender);
    }
    function PickWinner() public CheckAdmin{
        uint index = random_generator() % players.length;
        players[index].transfer(this.balance);
        winner_address = players[index];
        players = new address[](0);
    }
    function random_generator() public view returns(uint){
        return uint(keccak256(block.difficulty, now, players));
    }
    function getPlayer() public view returns(address[]){
        return players;
    }
    function WinPrice() public view returns(uint)
    {
        return this.balance;
    }
    modifier CheckAdmin{
        require(msg.sender == manager);
        _;
    }

}
```

```
pragma solidity ^0.4.17;

contract lottary
{
    address public manager;
    address public winner_address;
    address[] public players;
    function lottary() public{
        manager = msg.sender;
    }
    function enter() public payable{
        require(msg.value > 0.01 ether);
        players.push(msg.sender);
    }
    function PickWinner() public CheckAdmin{
        uint index = random_generator() % players.length;
        players[index].transfer(this.balance);
        winner_address = players[index];
        players = new address[](0);
    }
    function random_generator() public view returns(uint){
        return uint(keccak256(block.difficulty, now, players));
    }
    function getPlayer() public view returns(address[]){
        return players;
    }
    function WinPrice() public view returns(uint)
    {
        return this.balance;
    }
    modifier CheckAdmin{
        require(msg.sender == manager);
        _;
    }

}
```

```
pragma solidity ^0.4.17;

contract lottary
{
    address public manager;
    address public winner_address;
    address[] public players;
    function lottary() public{
        manager = msg.sender;
    }
    function enter() public payable{
        require(msg.value > 0.01 ether);
        players.push(msg.sender);
    }
    function PickWinner() public CheckAdmin{
        uint index = random_generator() % players.length;
        players[index].transfer(this.balance);
        winner_address = players[index];
        players = new address[](0);
    }
    function random_generator() public view returns(uint){
        return uint(keccak256(block.difficulty, now, players));
    }
    function getPlayer() public view returns(address[]){
        return players;
    }
    function WinPrice() public view returns(uint)
    {
        return this.balance;
    }
    modifier CheckAdmin{
        require(msg.sender == manager);
        _;
    }

}
```