Рабочие материалы к практикуму по разработке смарт-контрактов

Светлана Русова
smartcontracts.engineer

СМАРТ-КОНТРАКТЫ

1. «Hello, world»
2. «Визитка»
3. «Адресная книга»
4. Контракт токена ERC20
5. Наследование и модификаторы
6. Проверка даты начала и завершения ICO
7. Обработка исключений
8. Использование библиотеки Zeppelin
9. Использование SaleAgent
10. Получаем актуальный курс ETH/USD
11. Баунти и токены для команды
12. SoftCap
13. Сжигание токенов
14. Примеры тестов

http://smartcontracts.engineer

HELLO, WORLD

```
pragma solidity ^0.4.16;
contract HelloWorld {
    string wellcomeString = "Hello, world!";
    function getData() constant returns (string) {
        return wellcomeString;
    }
    function setData(string newData) {
        wellcomeString = newData;
    }
}
```

ВИЗИТКА

```
pragma solidity ^0.4.16;
contract BusinessCard {
    string name;
    uint age;
    function getName() constant returns (string) {
        return name;
    }
    function setName(string newName) {
        name = newName;
    }
    function getAge() constant returns (uint) {
        return age;
    }
    function setAge(uint newAge) {
        age = newAge;
    }
}
```

АДРЕСНАЯ КНИГА

```
pragma solidity ^0.4.16;
contract BusinessCard {
    mapping (bytes32 => string) data;
    function setData(string key, string value) {
        data[sha3(key)] = value;
    }
    function getData(string key) constant returns(string) {
        return data[sha3(key)];
    }
}
```

КОНТРАКТ ТОКЕНА ERC20

```
pragma solidity ^0.4.13;

contract SimpleTokenCoin {

    string public constant name = "Simple Coint Token";

    string public constant symbol = "SCT";

    uint32 public constant decimals = 18;

    uint public totalSupply = 0;

    mapping (address => uint) balances;

    function balanceOf(address _owner) constant returns (uint balance) {
        return balances[_owner];
    }

    function transfer(address _to, uint _value) returns (bool success) {
        balances[msg.sender] -= _value;
        balances[_to] += _value;
        Transfer(msg.sender, _to, _value);
        return true;
    }

    function transferFrom(address _from, address _to, uint _value) returns (bool success) {
        return true;
    }

    function approve(address _spender, uint _value) returns (bool success) {
        return false;
    }

    function allowance(address _owner, address _spender) constant returns (uint remaining) {
        return 0;
    }

    event Transfer(address indexed _from, address indexed _to, uint _value);

    event Approval(address indexed _owner, address indexed _spender, uint _value);

}
```

ОБРАБОТКА ИСКЛЮЧЕНИЙ

```
pragma solidity ^0.4.16;
contract Ownable {
    address owner;

    function Ownable() {
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(msg.sender == owner);
        _;
    }

    function transferOwnership(address newOwner) onlyOwner {
        owner = newOwner;
    }

}
```

НАСЛЕДОВАНИЕ И МОДИФИКАТОРЫ В ТОКЕНЕ ERC20

```
pragma solidity ^0.4.13;
contract Ownable {
    address owner;

    function Ownable() {
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(msg.sender == owner);
        _;
    }

    function transferOwnership(address newOwner) onlyOwner {
        owner = newOwner;
    }
}
contract SimpleTokenCoin is Ownable {
string public constant name = "Simple Coint Token";
string public constant symbol = "SCT";
uint32 public constant decimals = 18;
uint public totalSupply = 0;
mapping (address => uint) balances;
mapping (address => mapping(address => uint)) allowed;

    function mint(address _to, uint _value) onlyOwner {
        assert(totalSupply + _value >= totalSupply && balances[_to] + _value >=
balances[_to]);
        balances[_to] += _value;
        totalSupply += _value;
    }

    function balanceOf(address _owner) constant returns (uint balance) {
        return balances[_owner];
    }

    function transfer(address _to, uint _value) returns (bool success) {
        if(balances[msg.sender] >= _value && balances[_to] + _value >= balances[_to]) {
            balances[msg.sender] -= _value;
            balances[_to] += _value;
            Transfer(msg.sender, _to, _value);
            return true;
        }
        return false;
    }

    function transferFrom(address _from, address _to, uint _value) returns (bool success) {
        if( allowed[_from][msg.sender] >= _value &&
            balances[_from] >= _value
            && balances[_to] + _value >= balances[_to]) {
            allowed[_from][msg.sender] -= _value;
            balances[_from] -= _value;
            balances[_to] += _value;
            Transfer(_from, _to, _value);
            return true;
        }
        return false;
    }

    function approve(address _spender, uint _value) returns (bool success) {
        allowed[msg.sender][_spender] = _value;
        Approval(msg.sender, _spender, _value);
        return true;
    }

    function allowance(address _owner, address _spender) constant returns (uint remaining)
{
        return allowed[_owner][_spender];
    }

    event Transfer(address indexed _from, address indexed _to, uint _value);

    event Approval(address indexed _owner, address indexed _spender, uint _value);

}
```

ПРОВЕРКА ДАТЫ НАЧАЛА И ОКОНЧАНИЯ ICO

```
contract Crowdsale {

    address owner;

    SimpleTokenCoin public token = new SimpleTokenCoin();

    uint start = 1508976000; //26.10.2017 00:00:00

    uint period = 28;

    function Crowdsale() {
        owner = msg.sender;
    }

    function() external payable {
        require(now > start && now < start + period*24*60*60);
        owner.transfer(msg.value);
        token.mint(msg.sender, msg.value);
    }

}
```