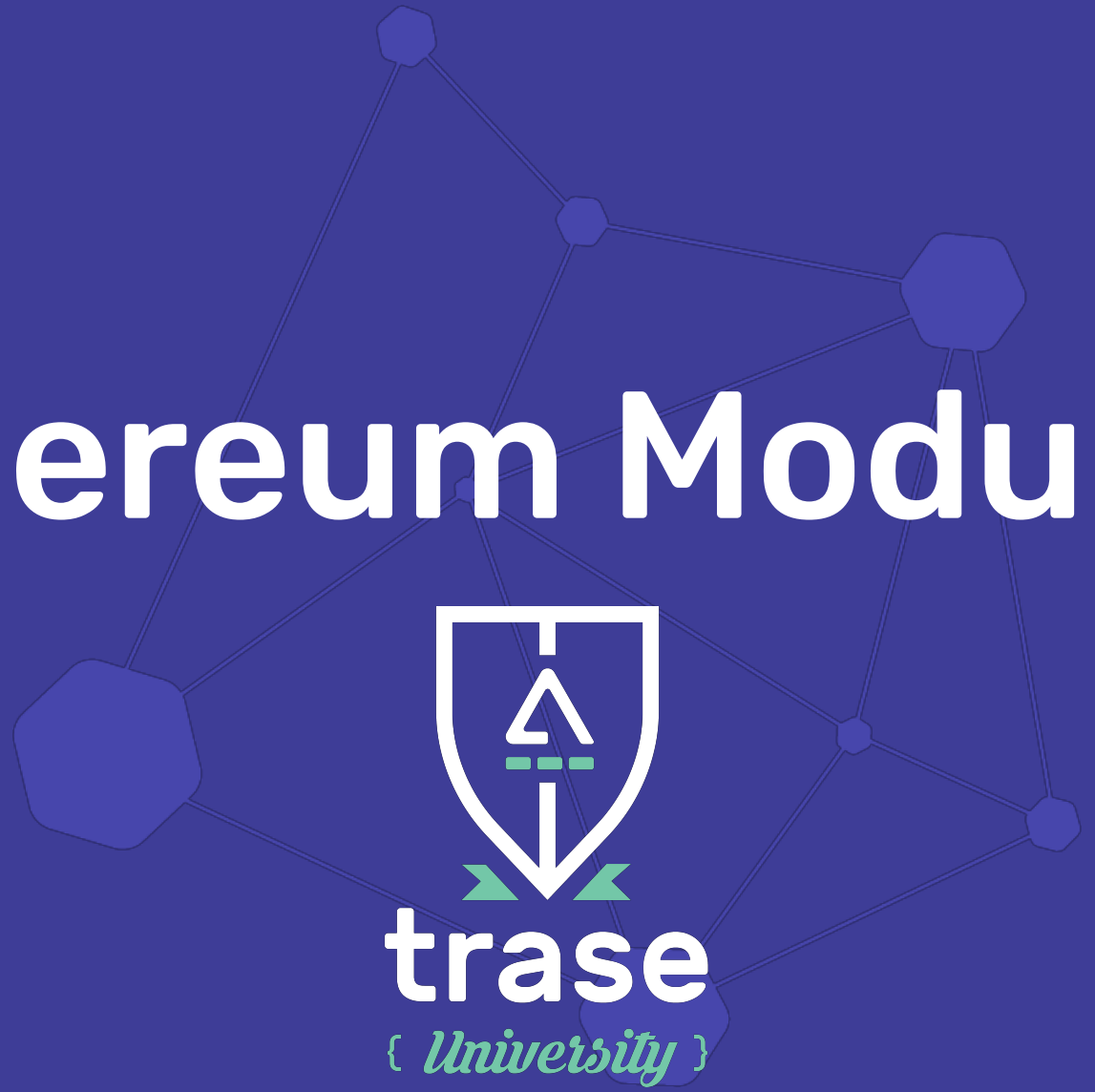
Ethereum Module II



Overview

Game on the Blockchain w/ CryptoZombies

Complete overview of Solidity

trAse

Creation of a Smart Contract: Hands on Solidity

Become a new kid on the blockchain.

trAse

Game on Blockchain

Cryptozombies -
Creation of Genetics

trase

Genetics - implementation

- 16-digit integer
  - ex: 8356281049284737

7 heads —> 83 % 7 + 1 = 7

9 eyes —> 56 % 9 + 1 = 3

...

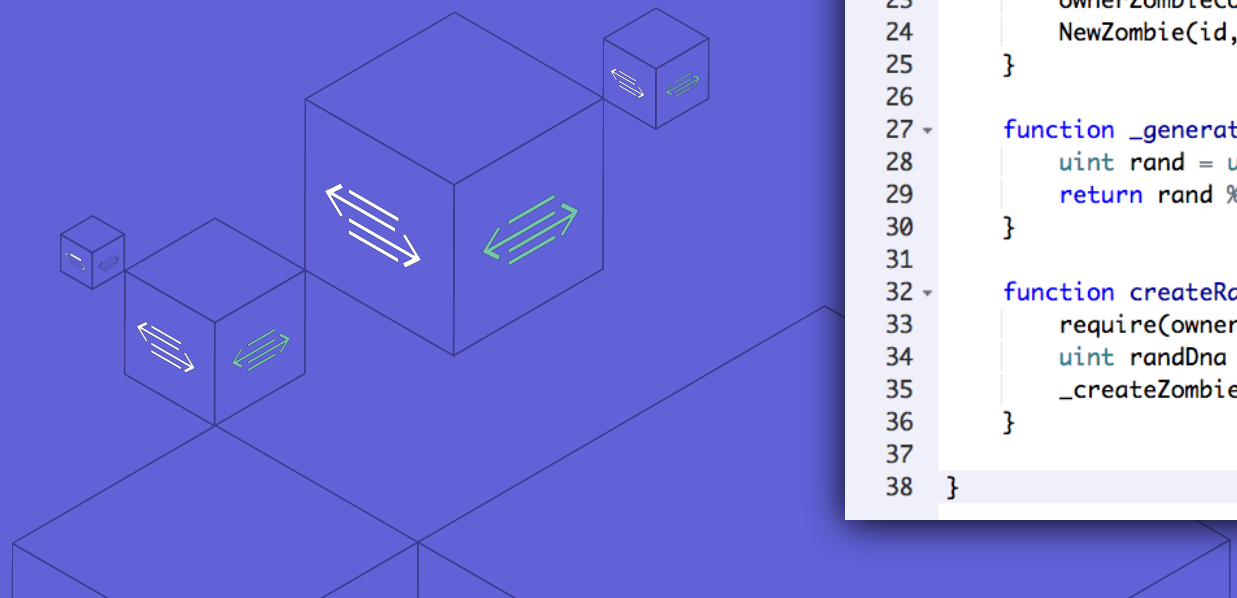
trase

End result

- Going through each concept

—> Remix IDE

https://remix.ethereum.org/



```
pragma solidity ^0.4.19;

contract ZombieFactory {

    event NewZombie(uint zombieId, string name, uint dna);

    uint dnaDigits = 16;
    uint dnaModulus = 10 ** dnaDigits;

    struct Zombie {
        string name;
        uint dna;
    }

    Zombie[] public zombies;

    mapping (uint => address) public zombieToOwner;
    mapping (address => uint) ownerZombieCount;

    function _createZombie(string _name, uint _dna) private {
        uint id = zombies.push(Zombie(_name, _dna)) - 1;
        zombieToOwner[id] = msg.sender;
        ownerZombieCount[msg.sender]++;
        NewZombie(id, _name, _dna);
    }

    function _generateRandomDna(string _str) private view returns (uint) {
        uint rand = uint(keccak256(_str));
        return rand % dnaModulus;
    }

    function createRandomZombie(string _name) public {
        require(ownerZombieCount[msg.sender] == 0);
        uint randDna = _generateRandomDna(_name);
        _createZombie(_name, randDna);
    }

}
```

Build the contract

- Contract keyword
- Define pragma version
  - Current version: **^0.4.24**

```
1  pragma solidity ^0.4.24;
2
3  contract ZombieFactory {
4      // start here
5  }
6
7
```

trAse

Variables and whole numbers

- Unsigned integer
- uint vs int?
- uint size - 256.. 32, 16, 8
- Typecasting: uint != int

```
1  pragma solidity ^0.4.24;
2
3  contract ZombieFactory {
4
5      uint dnaDigits = 16;
6      uint dnaModulus = 10 ** dnaDigits;
7
8      // start here
9  }
10
11
```

trAse

Structs

- Complex data structures
- Similar to objects

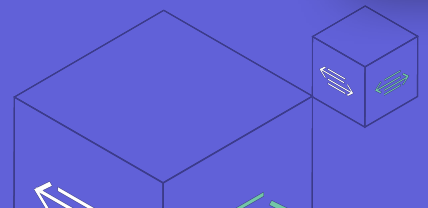
```
1  pragma solidity ^0.4.19;
2
3  contract ZombieFactory {
4
5      uint dnaDigits = 16;
6      uint dnaModulus = 10 ** dnaDigits;
7
8      struct Zombie {
9          uint dna;
10         string name;
11     }
12
13     Zombie[] public zombies;
14
15     // start here
16
17 }
18
```

trÆse

Functions

- Function modifiers
  - **Public/Private**
  - View
  - Pure

```
1  pragma solidity ^0.4.24;
2
3  contract ZombieFactory {
4
5      //declare our event here
6
7      uint dnaDigits = 16;
8      uint dnaModulus = 10 ** dnaDigits;
9
10     struct Zombie {
11         string name;
12         uint dna;
13     }
14
15     Zombie[] public zombies;
16
17     function _createZombie(string _name, uint _dna) private {
18         zombies.push(Zombie(_name, _dna));
19         // and fire it here
20     }
21 }
22
23
```



trÆse

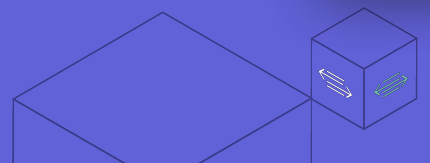
Functions

- Function modifiers
  - Public/Private
  - **View**
  - Pure

Hashing

- Keccak256

```
1  pragma solidity ^0.4.24;
2
3  contract ZombieFactory {
4
5      //declare our event here
6
7      uint dnaDigits = 16;
8      uint dnaModulus = 10 ** dnaDigits;
9
10     struct Zombie {
11         string name;
12         uint dna;
13     }
14
15     Zombie[] public zombies;
16
17     function _createZombie(string _name, uint _dna) private {
18         zombies.push(Zombie(_name, _dna));
19         // and fire it here
20     }
21
22     function _generateRandomDna(string _str) private view returns (uint){
23         uint rand = uint(keccak256(_str));
24         return rand % dnaModulus;
25     }
26 }
```



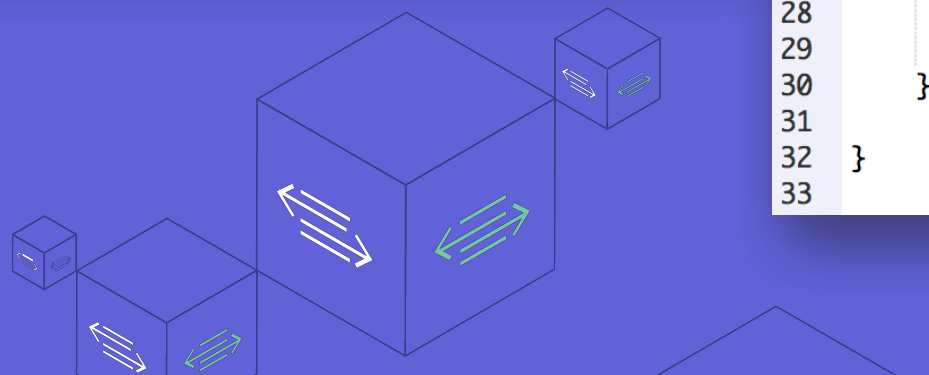
trÆse

Functions

- Function modifiers
  - Public/Private
  - View
  - **Pure**

```
function mergeZombies(uint _dna1, uint _dna2) public pure returns (uint){
    return (_dna1 + _dna2);
}
```

```
1  pragma solidity ^0.4.24;
2
3  contract ZombieFactory {
4
5      //declare our event here
6
7      uint dnaDigits = 16;
8      uint dnaModulus = 10 ** dnaDigits;
9
10     struct Zombie {
11         string name;
12         uint dna;
13     }
14
15     Zombie[] public zombies;
16
17     function _createZombie(string _name, uint _dna) private {
18         zombies.push(Zombie(_name, _dna));
19         // and fire it here
20     }
21
22     function _generateRandomDna(string _str) private view returns (uint){
23         uint rand = uint(keccak256(_str));
24         return rand % dnaModulus;
25     }
26
27     function createRandomZombie(string _name) public {
28         uint randDna = _generateRandomDna(_name);
29         _createZombie(_name, randDna);
30     }
31
32 }
33
```



trAse

Events

- React to events that happen in the contract
    - User feedback – by using listeners

```
pragma solidity ^0.4.24;

contract ZombieFactory {

    event NewZombie(uint zombieId, string name, uint dna);

    uint dnaDigits = 16;
    uint dnaModulus = 10 ** dnaDigits;

    struct Zombie {
        string name;
        uint dna;
    }

    Zombie[] public zombies;

    function _createZombie(string _name, uint _dna) private {
        zombies.push(Zombie(_name, _dna));
        // and fire it here
    }

    function _generateRandomDna(string _str) private view returns (uint){
        uint rand = uint(keccak256(_str));
        return rand % dnaModulus;
    }

    function createRandomZombie(string _name) public {
        uint randDna = _generateRandomDna(_name);
        _createZombie(_name, randDna);
    }

}

```

trÆse

Mapping – Address

- Mapping - key/value store
- msg.sender - invoker address

```
pragma solidity ^0.4.24;

contract ZombieFactory {

    event NewZombie(uint zombieId, string name, uint dna);

    uint dnaDigits = 16;
    uint dnaModulus = 10 ** dnaDigits;

    struct Zombie {
        string name;
        uint dna;
    }

    Zombie[] public zombies;

    mapping (uint => address) public zombieToOwner;
    mapping (uint => uint) ownerZombieCount;

    function _createZombie(string _name, uint _dna) private {
        uint id = zombies.push(Zombie(_name, _dna)) - 1;
        zombieToOwner[id] = msg.sender; // <--
        ownerZombieCount[msg.sender]++; // <--
        NewZombie(id, _name, randDna);
    }

    function _generateRandomDna(string _str) private view returns (uint){
        uint rand = uint(keccak256(_str));
        return rand % dnaModulus;
    }

    function createRandomZombie(string _name) public {
        uint randDna = _generateRandomDna(_name);
        _createZombie(_name, randDna);
    }
}
```

trÆse

If-Else..?

- Require
- Assert
- Throw

trAse

```
pragma solidity ^0.4.24;

contract ZombieFactory {

    event NewZombie(uint zombieId, string name, uint dna);

    uint dnaDigits = 16;
    uint dnaModulus = 10 ** dnaDigits;

    struct Zombie {
        string name;
        uint dna;
    }

    Zombie[] public zombies;

    mapping (uint => address) public zombieToOwner;
    mapping (uint => uint) ownerZombieCount;

    function _createZombie(string _name, uint _dna) private {
        uint id = zombies.push(Zombie(_name, _dna)) - 1;
        zombieToOwner[id] = msg.sender; // <--
        ownerZombieCount[msg.sender]++; // <--
        NewZombie(id, _name, randDna);
    }

    function _generateRandomDna(string _str) private view returns (uint){
        uint rand = uint(keccak256(_str));
        return rand % dnaModulus;
    }

    function createRandomZombie(string _name) public {
        require(ownerZombieCount[msg.sender] == 0); // <--
        uint randDna = _generateRandomDna(_name);
        _createZombie(_name, randDna);
    }
}
```

Small Demo

trAse

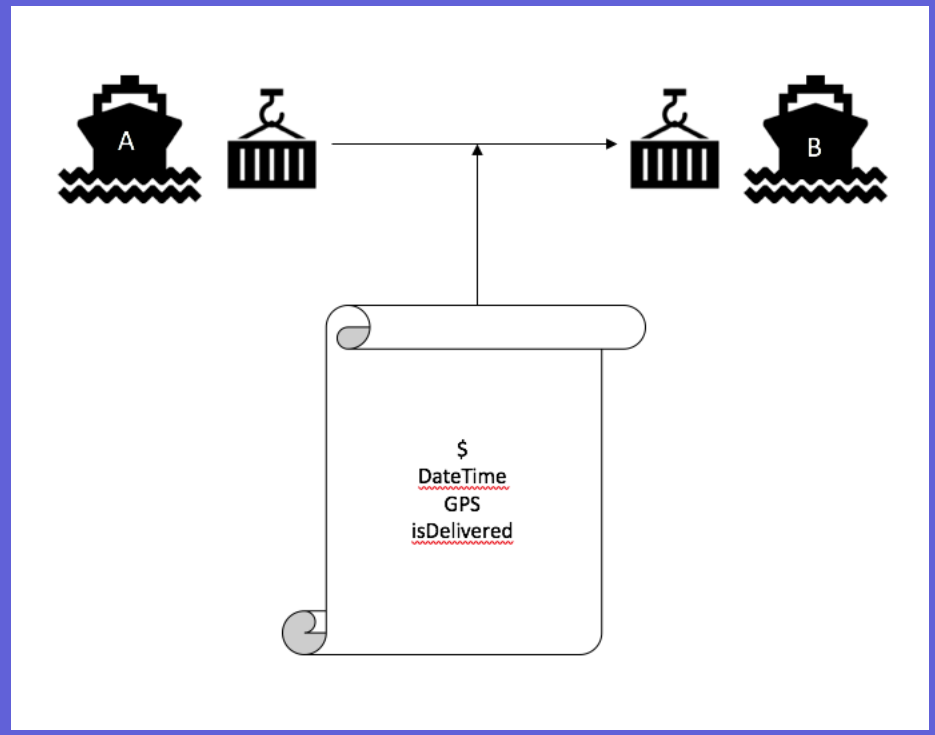
Questions?

trAse


trAse

Case

Ethereum



trase

Case Solution

trAse