Intro to Ethereum Smart Contracts

by Harrison and John

Ᵽitt €rypto

Pizza Sponsor

- https://lucylabs.io/
- Cryptocurrency Merchant Bank

lucylabs

Pitt Crypto

Overview

1. Installing Metamask / Ropsten faucet
2. Basic Solidity programming
3. Simple storage contract
4. Remix IDE
5. Token contract
6. Learning Resources

1 MetaMask

Installing Metamask

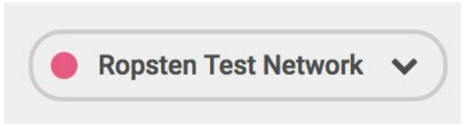
https://metamask.io/

Chrome/Firefox extension that allows you to interact with the Ethereum blockchain from your browser.

Get some test Ethereum

When testing smart contracts, you generally want to use a test network where the Ethereum are free.

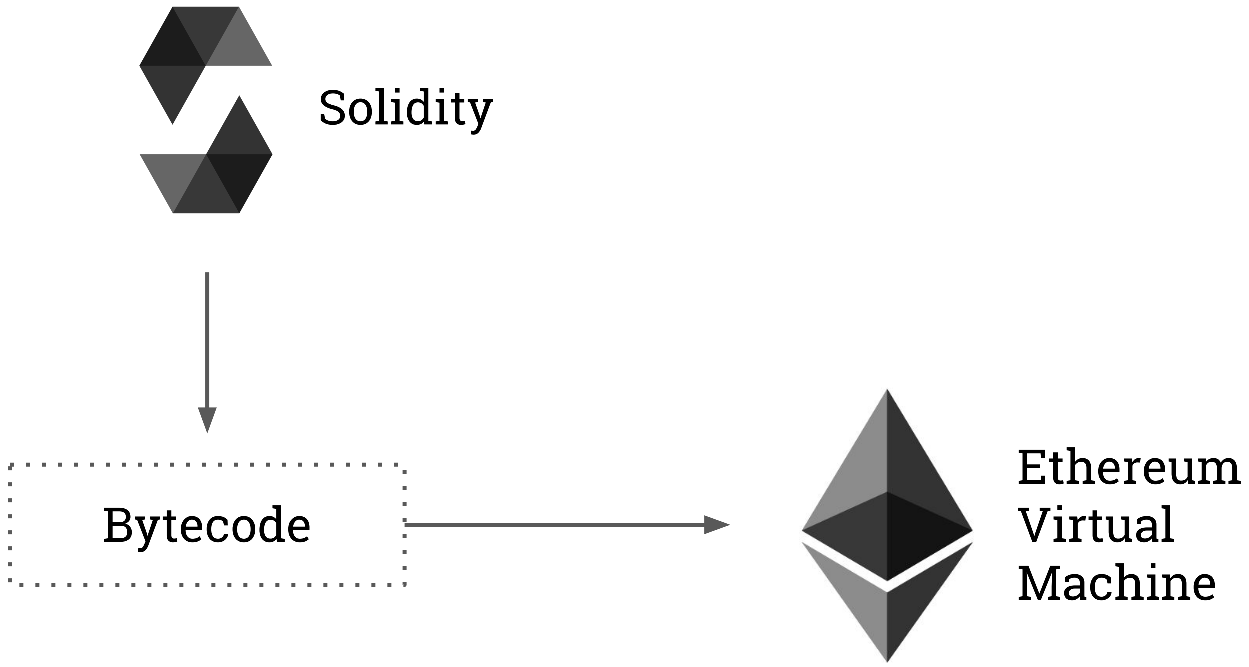
We are going to use the *Ropsten test network*.



1. In the top right hand corner, select Ropsten Test Network
2. Navigate to https://faucet.ropsten.be/ and enter your address to receive free test ethereum

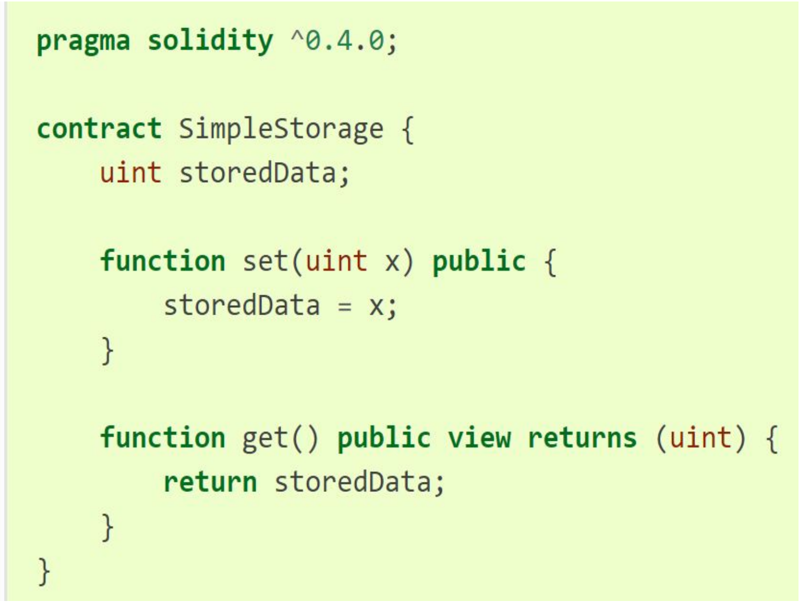
2 Solidity

Solidity is a high-level language



Solidity syntax

- Influenced by C++, Python and JavaScript



```
pragma solidity ^0.4.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

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

New ideas in Solidity/EVM

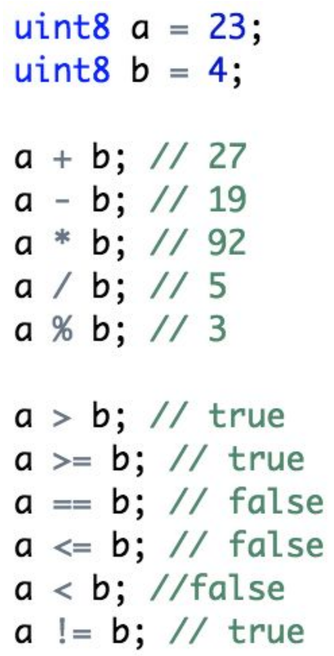
- Functions are called through transactions
- Programs can handle money directly (Ethereum)
- *more as we encounter them*

Solidity - Data Types

Solidity is statically typed

- **uint8**: 0 to 255
- **uint256**: 0 to 2^256 - 1
- **int8**: -128 to 127
- **bool**: true / false
- **address**: 0x06012c8cf97bead5deae237070f9587f8e7a266d
- **mapping (a => b)**: like a Java HashMap or Python dict

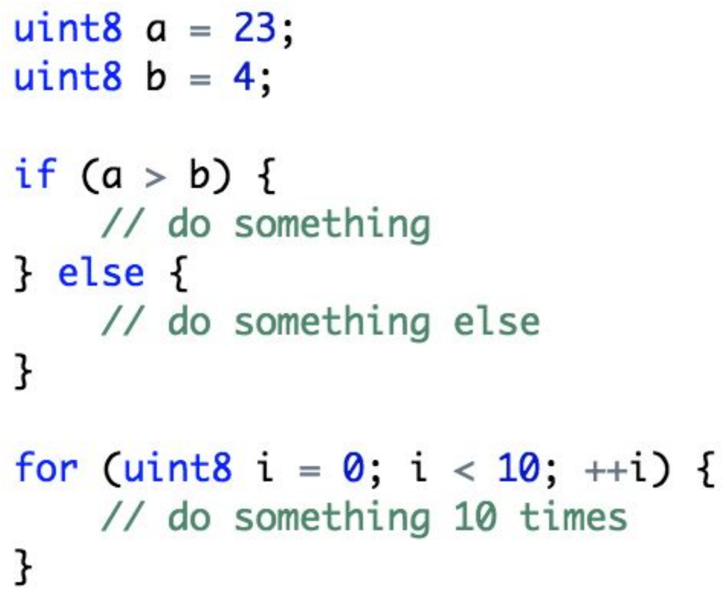
Solidity - Operators



```
uint8 a = 23;
uint8 b = 4;

a + b; // 27
a - b; // 19
a * b; // 92
a / b; // 5
a % b; // 3

a > b; // true
a >= b; // true
a == b; // false
a <= b; // false
a < b; //false
a != b; // true
```

Pitt Crypto

Solidity - Branching & Looping

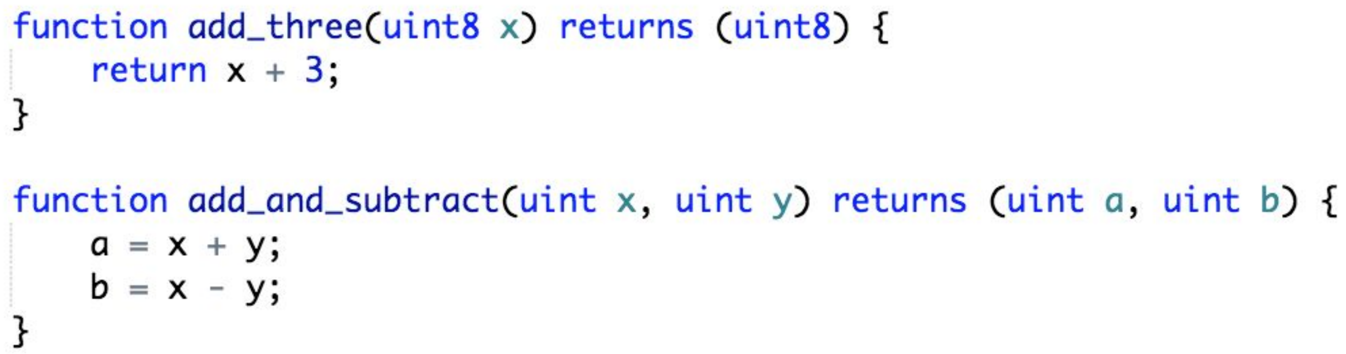


```
uint8 a = 23;
uint8 b = 4;

if (a > b) {
    // do something
} else {
    // do something else
}

for (uint8 i = 0; i < 10; ++i) {
    // do something 10 times
}
```

Pitt Crypto

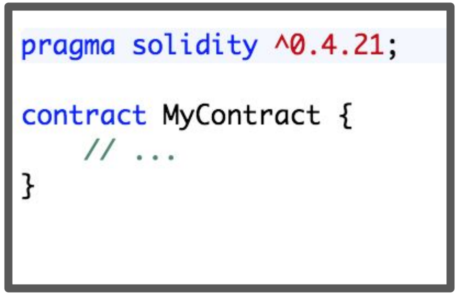
Solidity - Functions

- Functions take parameters
- Can have multiple (or zero) return values
- Modifiers - we will discuss later



```
function add_three(uint8 x) returns (uint8) {
    return x + 3;
}

function add_and_subtract(uint x, uint y) returns (uint a, uint b) {
    a = x + y;
    b = x - y;
}
```

Solidity - File

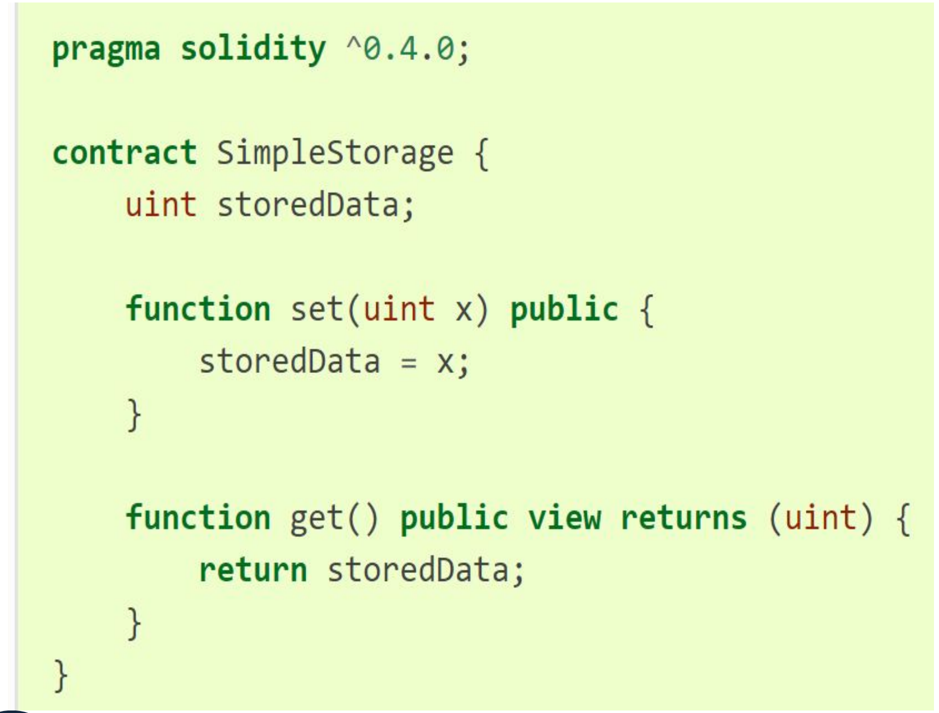


```
pragma solidity ^0.4.21;

contract MyContract {
    // ...
}
```

MyContract.sol

3 Simple Storage Contract

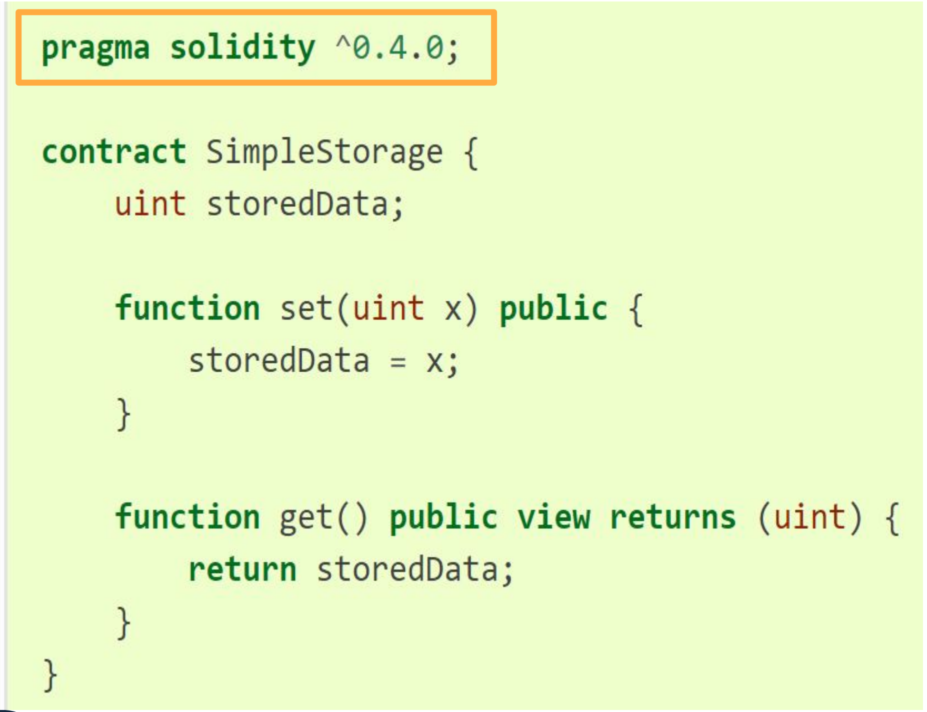
Dissecting a Smart Contract



```
pragma solidity ^0.4.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

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

- Very basic smart contract that allows any positive number to be stored to the Ethereum blockchain and then can be retrieved from blockchain

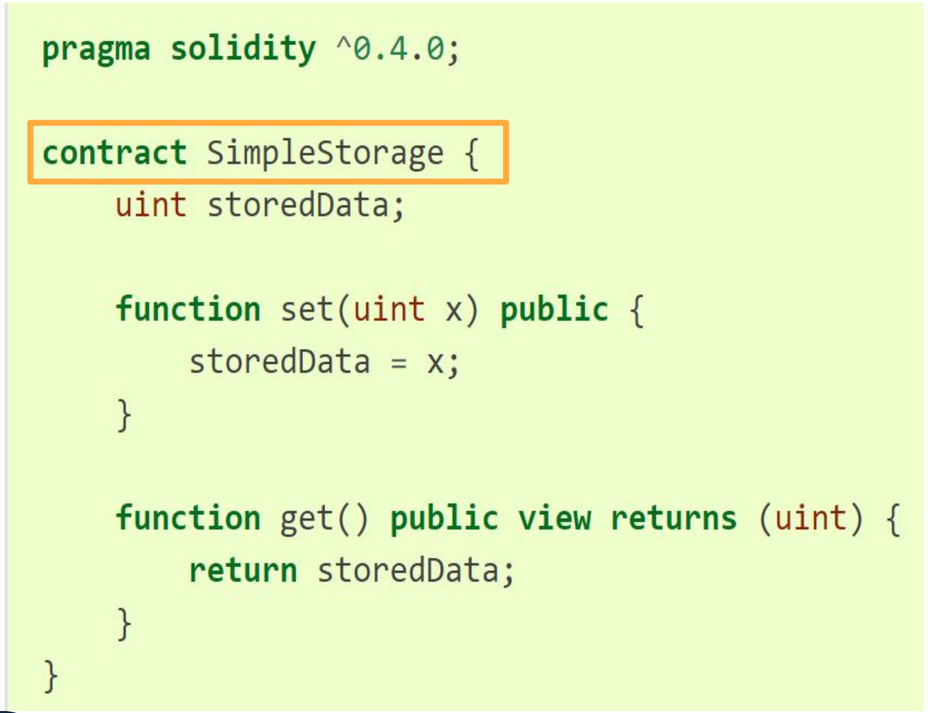
Dissecting a Smart Contract



```
pragma solidity ^0.4.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

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

- Indicates how the compiler will treat the source code
- Is required for every smart contract
- This is to ensure the code will not be compiled with a newer compiler where errors could occur

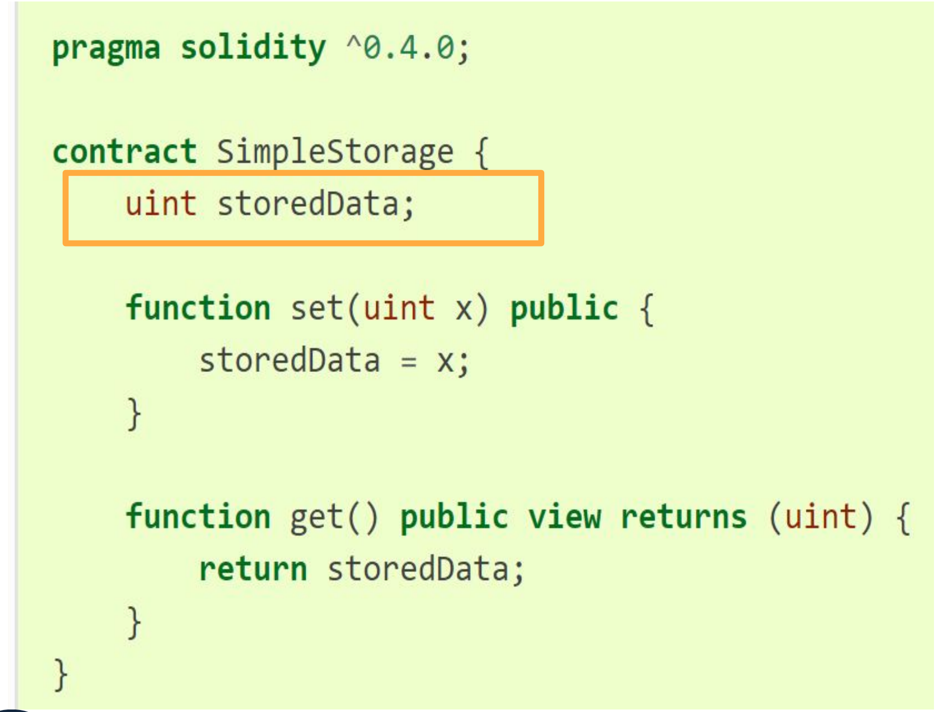
Dissecting a Smart Contract



```
pragma solidity ^0.4.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

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

- contract Name { } is also required for everything smart contract
- This denotes the name of the contract and all the data and functions inside of it

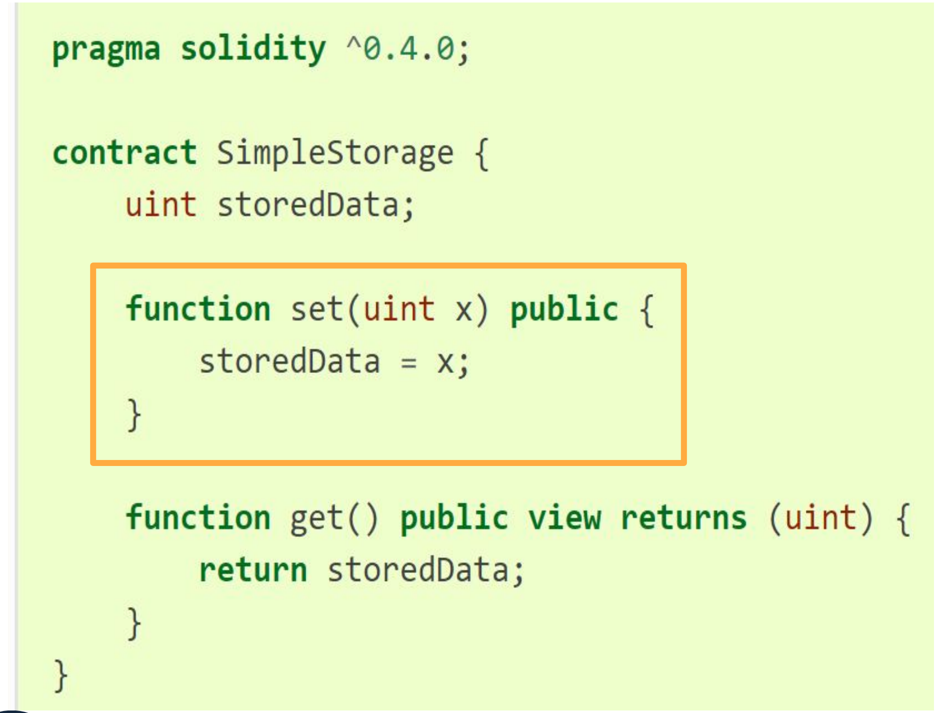
Dissecting a Smart Contract



```
pragma solidity ^0.4.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

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

- uint is a datatype that will store any data of any positive integer
- If no number is next to uint, it is automatically able to store up to 256 bits

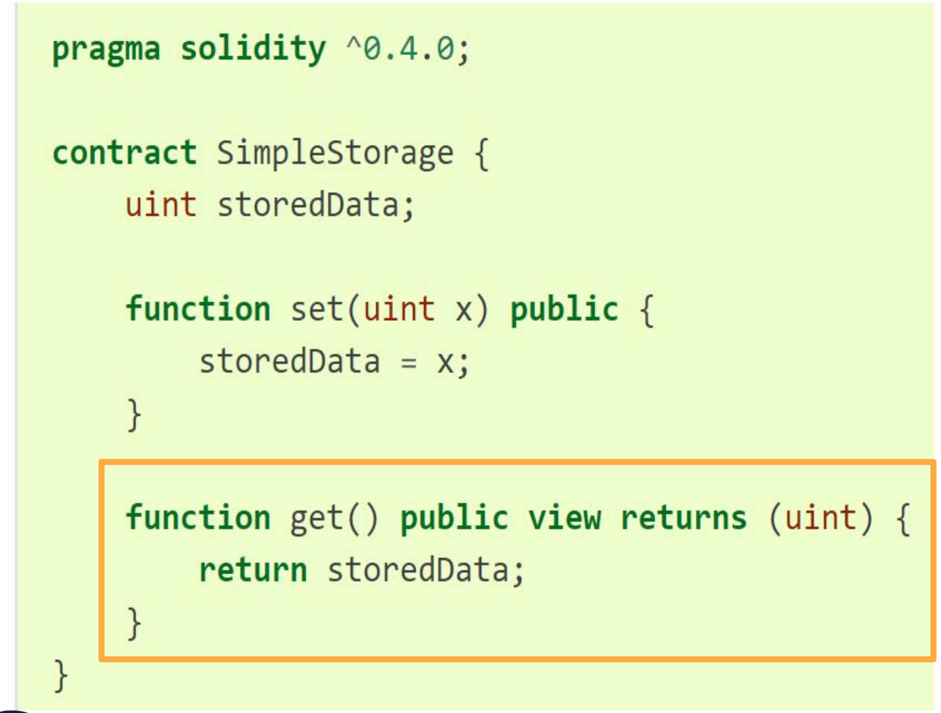
Dissecting a Smart Contract



```
pragma solidity ^0.4.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

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

- This declares a **function** called **set** that takes in an argument of a **uint** called **x**
- The previously declared variable **storedData** will be assigned the same value as the argument **x**
- If a function modifies of stores data in a contract, it has to be called on the blockchain through a transaction

Dissecting a Smart Contract



```
pragma solidity ^0.4.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

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

- This is a get function that will return the value of the storedData variable
- public means that any users or smart contracts can call this function
- view means no data is being changed on the Ethereum blockchain. View does not require gas to call
- returns (uint) will return a uint
- Must use a return statement if function has **returns** in it

4 Remix IDE

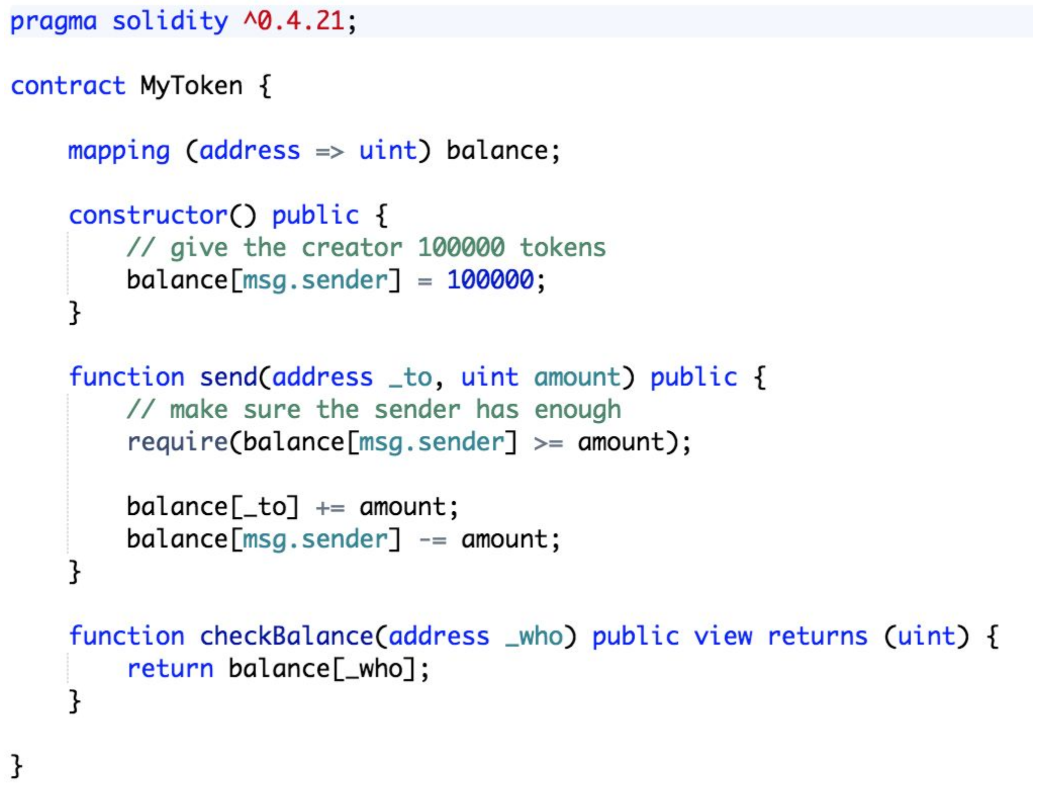
Remix IDE

https://remix.ethereum.org/

Platform to write and test smart contracts.

5 Token Contract

Token Contract



```solidity
pragma solidity ^0.4.21;

contract MyToken {

    mapping (address => uint) balance;

    constructor() public {
        // give the creator 100000 tokens
        balance[msg.sender] = 100000;
    }

    function send(address _to, uint amount) public {
        // make sure the sender has enough
        require(balance[msg.sender] >= amount);

        balance[_to] += amount;
        balance[msg.sender] -= amount;
    }

    function checkBalance(address _who) public view returns (uint) {
        return balance[_who];
    }

}
```

6 Learning Resources

Resources

Solidity Documentation (https://solidity.readthedocs.io)

CryptoZombies (https://cryptozombies.io/) - interactive tutorial to writing a game in Solidity