

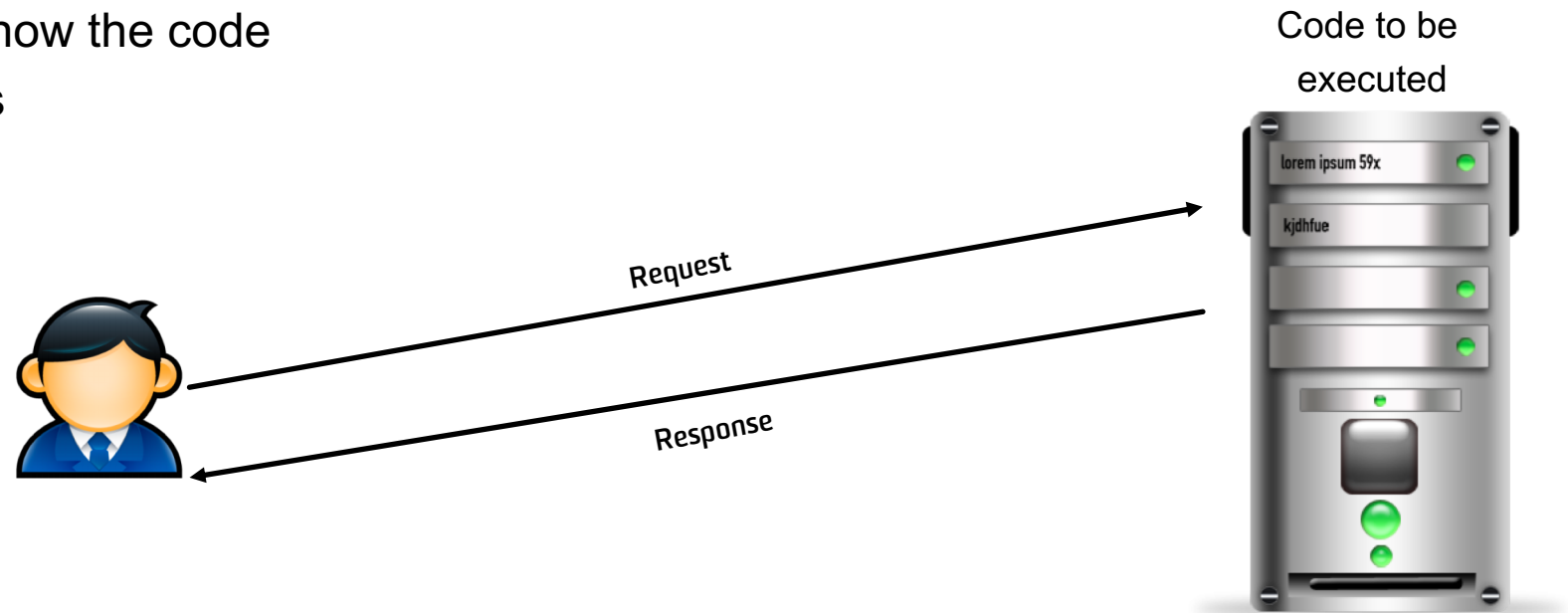


Learn Blockchain Programming

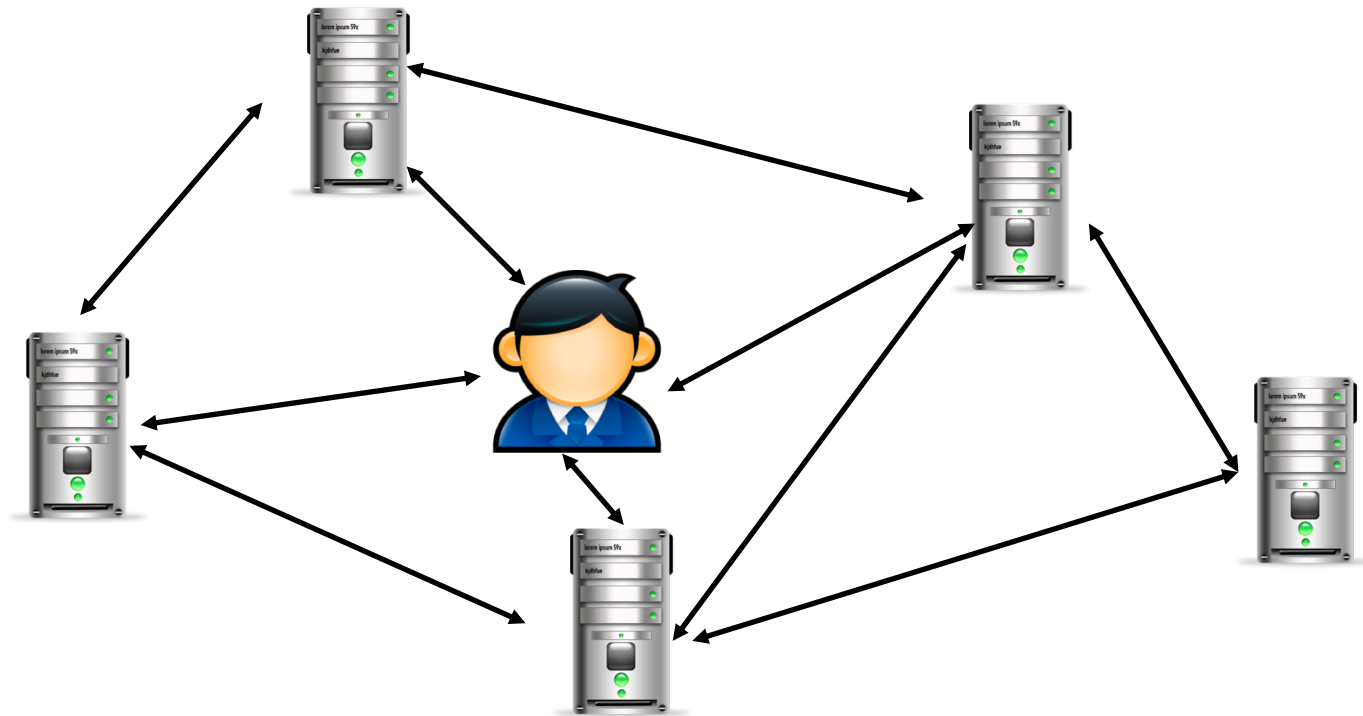
Ali Dorri

Traditional Programming

- Server runs the code
- User may or may not know the code
- Complicated algorithms
- Database



DApps: Distributed Applications



DApps: Distributed Applications

Open Source: All nodes have to verify the contract to verify transaction and state in blockchain.

Decentralized

Incentive: Normally use token/assets to fuel the code

Algorithm/Protocol

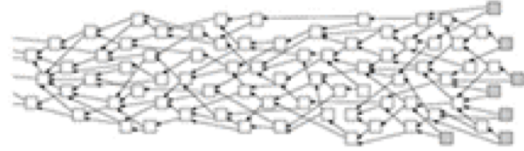
Immutable

Different Blockchains, Different Platforms



HYPERLEDGER

Tangle



IOTA

Blockchain



**Bitcoin
Ethereum**

- To store its transaction, each node has to verify two other transactions
- Self-scaling
- Reduce cost and delay



HYPERLEDGER

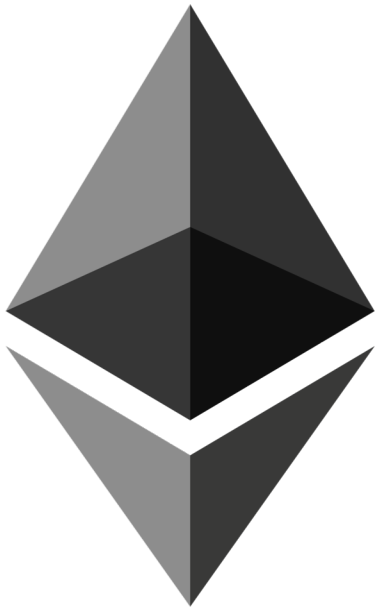
- Different blockchains
- Can be used to build blockchain and evaluate performance
- Open source

Solidity



- Influenced by C++, Java, and Python
- Runs on Ethereum Virtual Machine (EVM)
- EVM: The computer that all full nodes agree to run, i.e., runtime environment for smart contract
- All nodes perform the code execution
- Write Solidity code online in: <https://remix.ethereum.org>

Accounts



- Each account has an address:
 - External accounts: determined by public key
 - Smart contract account: creator address and nonce
- Each account has *balance* in Ether (in “Wei” to be exact, 1 ether is 10^{18} wei)

Gas

- Gas Limit
- Gas Price

$$\text{Transaction fee} = \text{Gas limit} * \text{Gas price}$$

Gas limit: Amount of liters of gas for a car

Gas price: Cost of liter of gas

Solidity



```
pragma solidity ^0.4.0;

contract SimpleStorage {
    uint storedData;

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

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

You can define *Structs* in solidity

```
struct Person {  
    uint age;  
    string name;  
}
```

You can define *Arrays* in solidity as well

```
// Array with a fixed length of 2 elements:  
uint[2] fixedArray;  
  
// another fixed Array, can contain 5 strings:  
string[5] stringArray;  
  
// a dynamic Array - has no fixed size, can keep growing:  
uint[] dynamicArray;
```



- **Define Function**

```
function setAge(string _name, uint _age) {  
  
}
```

Function visibility

- Public
 - Your contract can be called by anyone or any other contract
 - Default
 - Security issue



- Private

- Only functions within the contract can call the function

```
function setAge(string _name, uint _age) private {  
  
}
```

- Internal

- Similar to private, but accessible to contracts that inherit from this contract

- External

- Similar to public, but can “only” be called outside the contract

- Return value

```
string greeting = "What's up";  
function sayHello() public returns (string) {  
    return greeting;  
}
```

- Function modifiers

- View: the function only views data, but not modifying any value, i.e., only read action is performed.
- Pure: the function does not access data in blockchain, i.e., the return value only depends on the input values.

```
function _multiply(uint a, uint b) private pure returns  
(uint) {  
    return a * b;  
}
```

- Mapping (key→value)

```
mapping (address => uint) public accountBalance;
```

```
mapping (uint => string) userIdToName;
```

- How to find out the address of the person who called the transaction?

```
msg.sender
```

- Running a function with a condition:

```
function sayHiToVitalik(string _name) public returns (string) {  
require(keccak256(_name) == keccak256("Ali"));  
return "Hi!";  
}
```


Contract inheritance

```
contract Doge {  
function catchphrase() public returns (string) {  
return "So Wow CryptoDoge";  
}}
```

```
contract BabyDoge is Doge {  
function anotherCatchphrase() public returns (string) {  
return "Such Moon BabyDoge";  
} }
```

```
import "./someothercontract.sol";
```

Ownable

- “Ownable” is a contract from OpenZeppelin.
- OpenZeppelin is a library for secure smart contract development.
- Has a modifier named “onlyOwner”

```
contract MyContract is Ownable {  
  
    function likeABoss() external onlyOwner {  
        LaughManiacally("Muahahahaha");  
    }  
}
```

<https://github.com/OpenZeppelin/openzeppelin-solidity>

Payable

- Marks a contract as payable

```
contract OnlineStore {  
  function buySomething() external payable {  
  
    require(msg.value == 0.001 ether);  
  
    transferThing(msg.sender);  
  }  
}
```

- Withdraw

```
uint itemFee = 0.001 ether;  
msg.sender.transfer(msg.value - itemFee);
```



What to do after writing the smart code?

To interact with a contract you need:

- 1.The address of the smart contract
- 2.The function you want to call, and
- 3.The variables you want to pass to that function.

JSON files

Infura

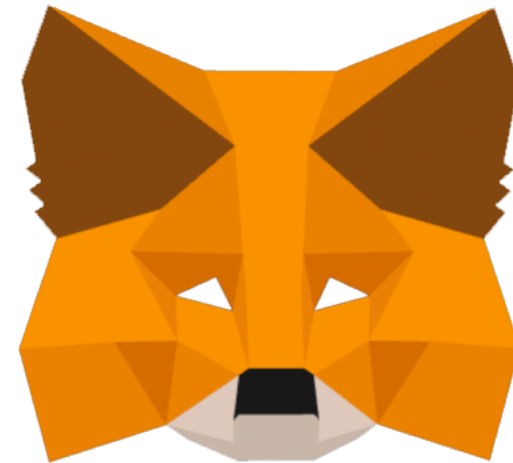
- Maintains a set of Ethereum nodes
- Connects you to the Ethereum
- Use the address in your API.

```
var web3 = new Web3(new Web3.providers.WebsocketProvider  
("wss://mainnet.infura.io/ws"));
```

<https://infura.io/>

Metamask

- Browser extension for chrome and Firefox
- Allows users to manage their Ethereum accounts while connecting to websites.
- As a developer, if you want the users to interact with your DApp using website, you need to make it metamask-compatible.
- Check if the user has installed metamask





Contract Application Binary Interface (ABI)

- Your contract in JSON format
- Clarifies how to call functions

Private Ethereum testnet

- **Install Node.js,**
 - `brew install node`
- **Install compiler**
 - `npm install -g solc`
- **Install Ethereum**
 - `brew tap ethereum/ethereum`
 - `brew install ethereum`

Create a
genesis.JSON file

```
{
  "config": {
    "chainId": 15,
    "homesteadBlock": 0,
    "eip155Block": 0,
    "eip158Block": 0
  },
  "difficulty": "1",
  "gasLimit": "2100000",
  "alloc":{
    "yourNewlyCreatedAccountAddressMustGoHere": {
      "balance": "300000"
    },
    "yourNewlyCreatedAccountAddressMustGoHere": {
      "balance": "400000"
    },
    "yourNewlyCreatedAccountAddressMustGoHere": {
      "balance": "500000"
    }
  }
}
```

Initialize the first node

```
geth --datadir ~/gethDataDir/ init genesis.json
```

Start the first node



```
geth --datadir ./myDataDir --networkid 1114 console 2>> myEth.log
```

Create account

```
personal.newAccount("<YOUR_PASSPHRASE>")
```

Check account

```
Eth.accounts
```



Checking account balance `Eth.getbalance(eth.coinbase/eth.accounts[0])`

Start mining `Miner.start(1)`

Add another node

```
geth --datadir ./peer2DataDir --networkid 1114 --port 30304  
console 2>> myEth2.log
```

Node address `admin.nodeInfo.enode`

Verify peers `Admin.peers`

How to create your own cryptocurrency (ICO)?





Token

- Token is a smart contract that follows some common rules and implements a standard set of functions.



“That's all Folks!”

Ping me at ali.dorri@unsw.edu.au