Solidity Tutorial
How to launch your own token “Hands on”

Joshi

Smart Contracts
Ether currency
Dapps (decentralized applications)
Blockchain infrastructure and principles
Secure, decentralized, tamper-proof
Understanding cryptography
Introduction to blockchain programming
Solidity variables, Solidity control structure, Solidity functions, Solidity inheritance, Solidity modifiers, Proxy contracts, Solidity events
Development frameworks: Truffle Framework
Web3 JavaScript API
DAO (decentralized autonomous organization)

Overview of decentralized platforms
Ethereum Blockchain App Platform
Creating the Smart Contract
Writing the Smart Contract functions
Storing dept registries
Deploying the Smart Contract
Ethereum Wallet
Holding and securing ether
Managing other crypto-assets
Creating your own cryptocurrency
Overview of tradeable digital token and coin APIs
Design & Issuing the cryptocurrency
Kickstarting a blockchain project
Initiating a trustless crowdsale
Building your own virtual organization

Goals for the day

Understanding the two tools: Metamask / Ether Wallet

Creating Wallet at metamask and receiving coin from individual and from Faucet

Creating a simple solidity program on remix

Metamask

MetaMask & Ether Wallet

Installing Metamask

An Extension

Getting fake ether from faucet

Installing Ether Wallet

Understanding the contract implementation

Problems we face in the open source universe

1. Installing metamask
2. Faucet not giving fake ether

Wallet problems (we are not using this method):

Ports Closed by Firewall for Ether Wallet

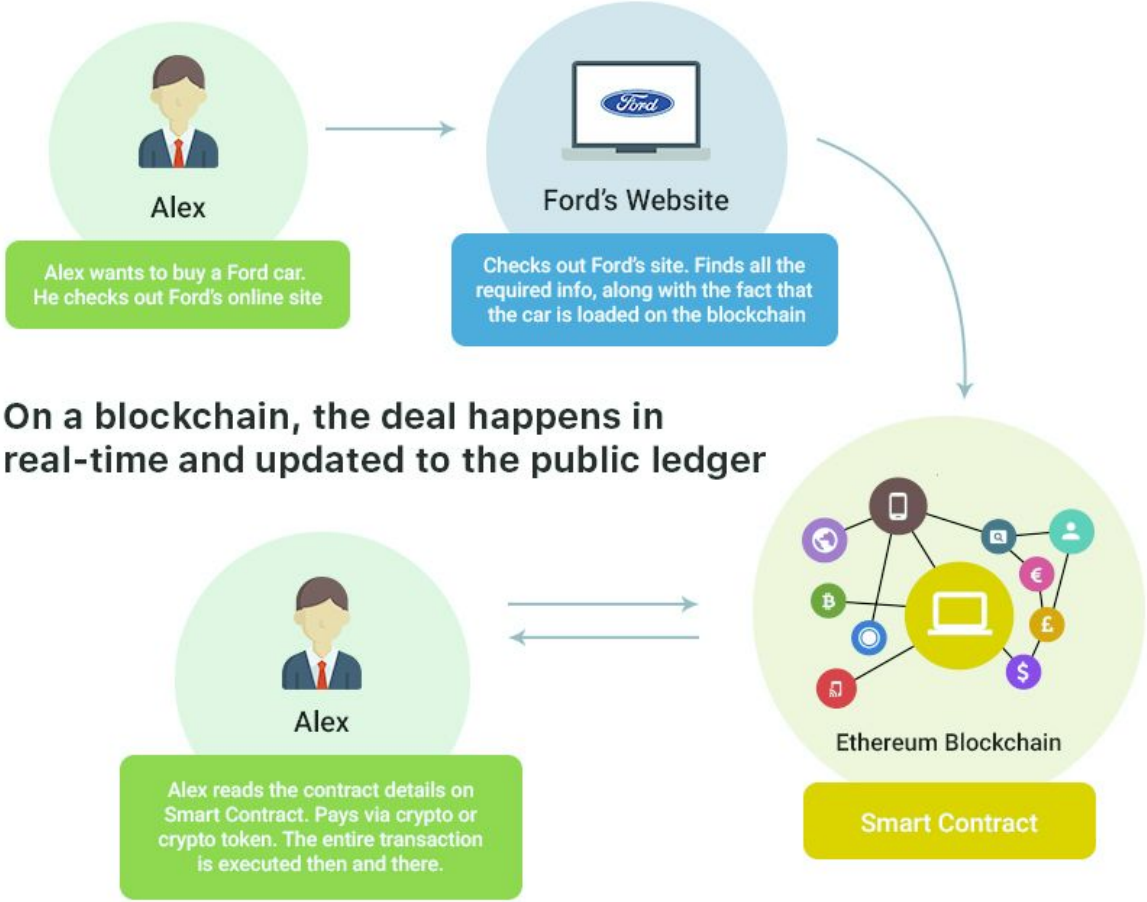
No Peer to download the blockchain for Rinkby on Ether Wallet

Smart Contract & Solidity

What elements do we need to create a contract?

1. Address (user and contract)
2. Functions (to do something)
3. Currency (which tokens and costs)

BUYING THE CAR ON ETHEREUM BLOCKCHAIN



Metamask settings

JavaScript VM

Execution environment does not connect to any node, everything is local and in memory only.

Injected Web3

Execution environment has been provided by Mist or similar provider.

Web3 Provider

Execution environment connects to node at localhost (or via IPC if available), transactions will be sent to the network and can cause loss of money or worse!

If this page is served via https and you access your node via http, it might not work. In this case, try cloning the repository and serving it via http.

Web3 Provider Endpoint: http://localhost:8545

Attach Transact Transact (Payable) Call

Toggle Details

DataContract.sol:DataContract 4191 bytes

At Address Create

Transaction cost: 1163338 g
Execution cost: 838534 gas.

DataContract.sol:DataContract at 0x692a70d2e424a56d2c6c27aa97d1a86395877b3a (memory)

users uint256

count **Value:** "0x0000000000000000000000000000000000000000000000000000000000000000"
Transaction cost: 21505 gas. (caveat)
Execution cost: 233 gas.
Decoded:
1. uint256: 0

owner **Value:** "0x000000000000000000000000ca35b7d915458ef540ade6068dfe2f44e8fa733c"
Transaction cost: 21729 gas. (caveat)
Execution cost: 457 gas.
Decoded:
1. address: 0xca35b7d915458ef540ade6068dfe2f44e8fa733c

DataContract.sol:DataContract 4191 bytes

At Address Create

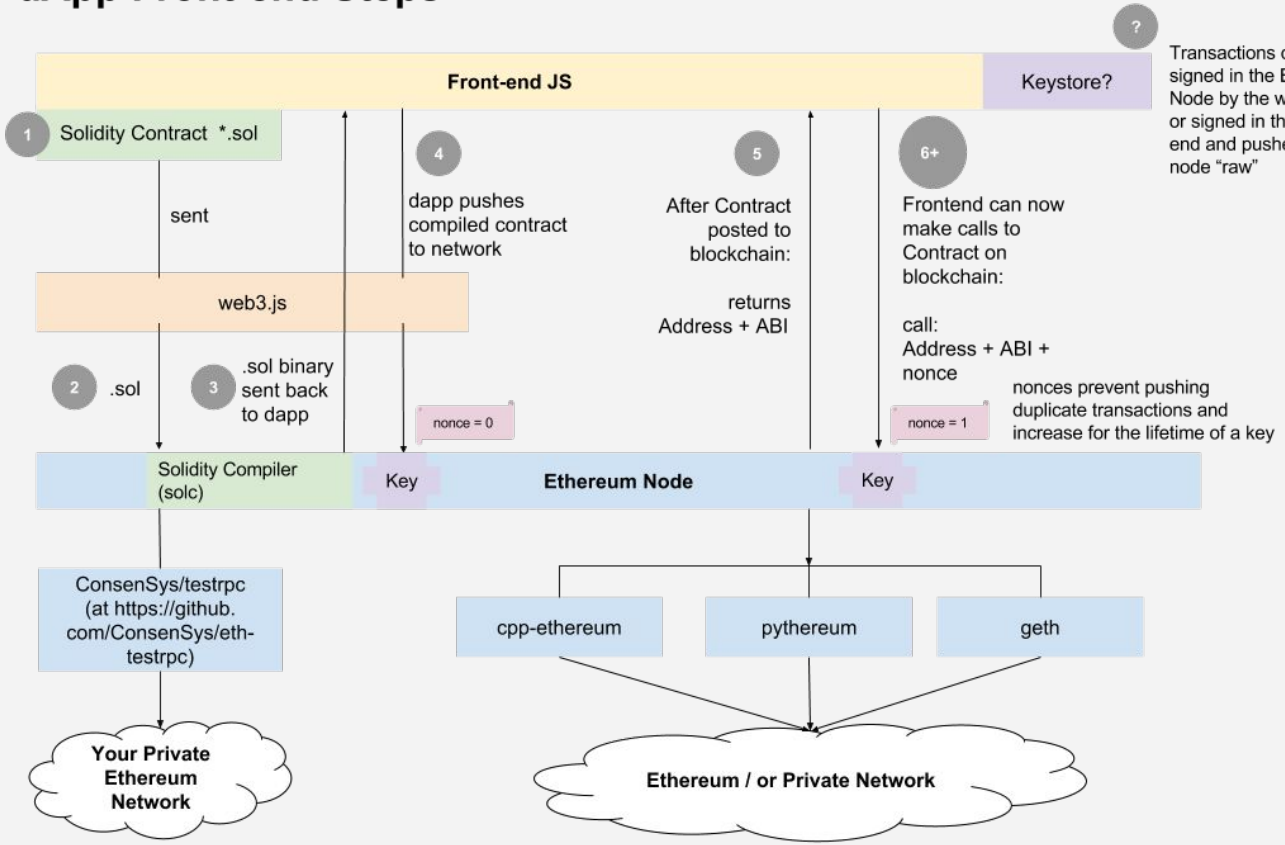
Bytecode 60606040525b33600060006101000a81

Interface [{"constant":true,"inputs":[],"name":"cour

Web3 deploy

```
var datacontract.sol:datacontra
var datacontract.sol:datacontra
  {
    from: web3.eth.accounts[0]
    data: '0x60606040525b33600
    gas: '4700000'
  }, function (e, contract){
   console.log(e, contract);
   if (typeof contract.address
        console.log('Contract
   }
 })
```

dApp Front-end Steps



A **Contract Creation Transaction** is shown in steps 1-5 at above.

An **Ether Transfer** or **Function Call Transaction** is assumed in step 6.

Implementing a new contract

This takes time so we will do this first and then practice solidity

Getting fake ether

Deploy using fake ether

https://www.ethereum.org/token

What goes inside a contract?

Contract

Address

Blockchain

Constants

Functions

First Ethereum Program to *set and get age and name*

Understanding Data types

Creating Functions

Identifying the arguments and return values

Set age vs get age

```
pragma solidity ^0.4.0;
contract MyFirstContract {
        string private name;
        uint private age;

function setName(string newName) {
        name = newName;      }

function getName() returns (string) {
        return name;       }

function setAge(uint newAge) {
        age = newAge;   }

function getAge() returns (uint) {
        return age; }
}
```

Public Constants in Solidity

```
pragma solidity ^0.4.0;

contract publiconst {

    string public constant symbol = "MFT";

    string public constant name = "My First Token";

    uint8 public constant decimals = 18; }
```

// What are the arguments in the functions below?

```
pragma solidity ^0.4.0;
contract MyFirstContract {
    string private name;
    uint private age;
    function setName(string newName) { name = newName;  }
    function getName() returns (string) {  return name; }
    function setAge(uint newAge) {     age = newAge;   }
    function getAge() returns (uint) {return age;  } }
```

```
pragma solidity ^0.4.2;
contract MyToken {
    /* This creates an array with all balances */
    mapping (address => uint256) public balanceOf;
    /* Initializes contract with initial supply tokens to the creator of the contract */
     function MyToken(
        uint256 initialSupplybyjoshi              ) public {
            initialSupplybyjoshi =121;
        balanceOf[msg.sender] = initialSupplybyjoshi;      // Give the creator all initial tokens
    }    /* Send coins */
    function transfer(address _to, uint256 _value) public {
        require(balanceOf[msg.sender] >= _value);           // Check if the sender has enough
        require(balanceOf[_to] + _value >= balanceOf[_to]); // Check for overflows
        balanceOf[msg.sender] -= _value;                    // Subtract from the sender
        balanceOf[_to] += _value;                           // Add the same to the recipient
 }}
```

https://rinkeby.etherscan.io/tx/0xd93370d9a11ddd54adee917afe6db57bf9553804a47

Etherscan RINKEBY
The Ethereum Block Explorer

RINKEBY (CLIQUE) TESTNET

Search by Address / T

HOME | BLOCKCHAIN | ACCOUNT | TOKEN

Transaction 0xd93370d9a11ddd54adee917afe6db57bf9553804a470dc9464894d9e36c5f10e

Home /

Overview

Transaction Information

TxHash:	0xd93370d9a11ddd54adee917afe6db57bf9553804a470dc9464894d9e36c5f10e
TxReceipt Status:	Success
Block Height:	2099390 (2 block confirmations)
TimeStamp:	32 secs ago (Apr-12-2018 07:21:33 PM +UTC)
From:	0x03f472f252e73a113368020d199208148325103a
To:	[Contract 0xce565f070120b91dc3c7701c88ae1d99ac676d4b Created]
Value:	0 Ether ($0.00)
Gas Limit:	256456
Gas Used By Txn:	256456
Gas Price:	0.000000001 Ether (1 Gwei)
Actual Tx Cost/Fee:	0.000256456 Ether ($0.000000)
Nonce:	0

Code

https://tinyurl.com/y9hs8ydn

Launching your Token

From the official page

Link: https://www.ethereum.org/token

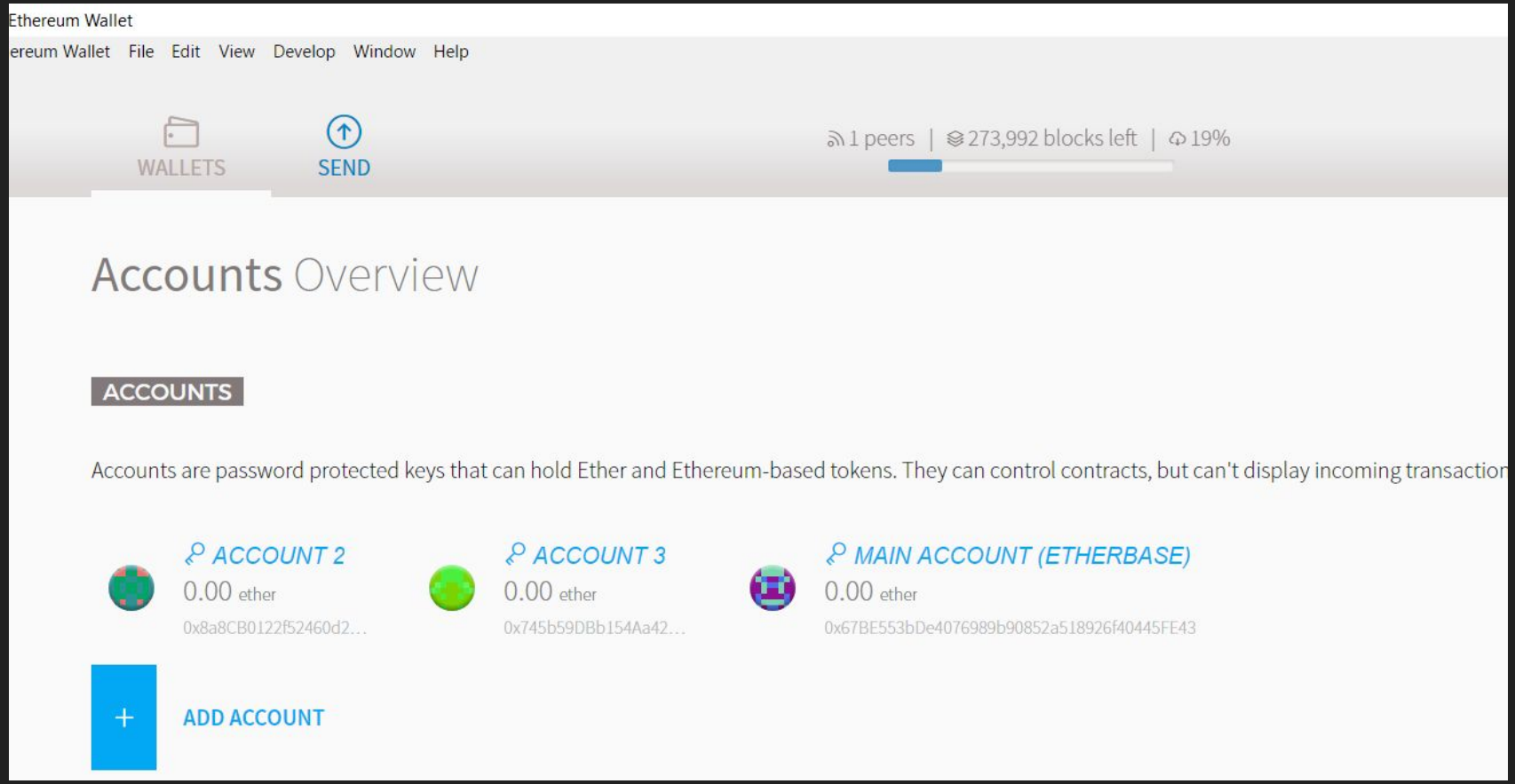
Two Tools to launch your coin

MetaMask is a bridge that allows you to visit the distributed web of tomorrow in your browser today.

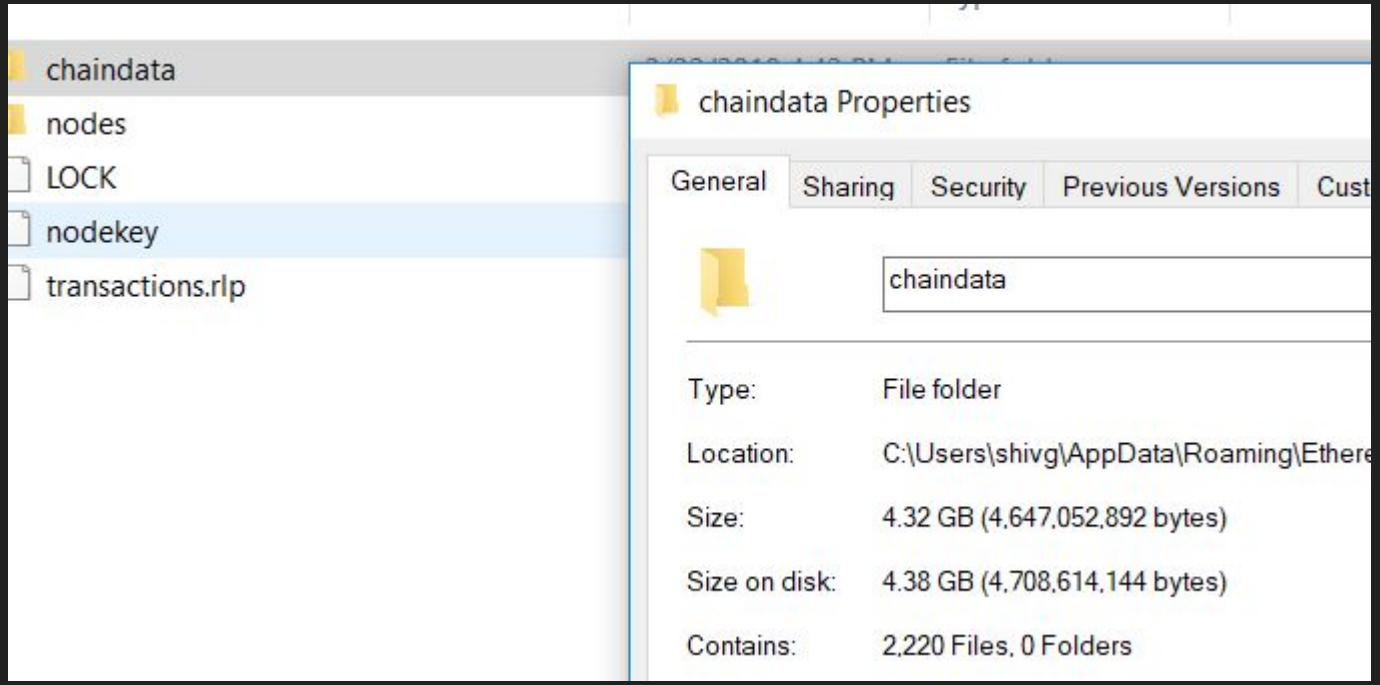
Ethereum Wallet

Run Ethereum dApps right in your browser without running a full Ethereum node

Ether Wallet



Chain Data


chaindata
nodes
LOCK
nodekey
transactions.rlp
chaindata Properties
General
Sharing
Security
Previous Versions
Cust
chaindata
Type:
File folder
Location:
C:\Users\shivg\AppData\Roaming\Ethere
Size:
4.32 GB (4,647,052,892 bytes)
Size on disk:
4.38 GB (4,708,614,144 bytes)
Contains:
2,220 Files, 0 Folders

construct.

```
mapping (uint => Customer) customers;

mapping (uint => Customer) customers;

customers[key];

public uint count = 0;
```

Program 2: Bank

Understanding Inheritance

Keywords

Interface

Contract X is interface

Libraries

Libraries

Libraries are similar to contracts, but their purpose is that they are deployed only once at a specific address and their code is reused using the DELEGATECALL (CALLCODE until Homestead) feature of the EVM.
This means that if library functions are called, their code is executed in the context of the calling contract, i.e. this points to the calling contract, and especially the storage from the calling contract can be accessed.
As a library is an isolated piece of source code, it can only access state variables of the calling contract if they are explicitly supplied (it would have no way to name them, otherwise).
Library functions can only be called directly (i.e. without the use of DELEGATECALL) if they do not modify the state (i.e. if they are view or pure functions), because libraries are assumed to be stateless.
In particular, it is not possible to destroy a library unless Solidity’s type system is circumvented.