



# ethereum

## vienna

Workshop  
Contract Development for Beginners



# Agenda

EVM Fundamentals

Mix IDE

Solidity Basics



# Fundamentals

## Transaction

- wraps a message

- signed by a private key

- only transactions appear in chain

- sets gasprice for all contained messages



# Fundamentals

Message

Sender

Recipient

Value (can be 0)

Data

Return Value

Gaslimit

Executes either completely or not at all



# Fundamentals

## Contract

160 bit address

Balance

EVM Bytecode

Runs at every received message

Has a persistent 256-to-256 bit storage

Private

Expensive

Can spawn new messages



# EVM

Stack machine

256 bit words

Has all the usual instructions plus

- block data, tx data, msg data, contract data access

- cryptographic functions

- message sending



# EVM

## Storage

- expensive

- persistent

## Memory

- cheaper

- byte-level access

## Stack

- inaccessible in solidity (except assembly)



EVM

Out of gas exception

Logs

for UIs

Light Clients

Logging

Self Destruct / Suicide





# ethereum

# Mix IDE

MyProject  
/Users/ralph/MyProject

contract.sol

index.html

contract.sol

1 //Sample contract  
2 contract Sample  
3 {  
4     uint value;  
5     function Sample(uint v) {  
6         value = v;  
7     }  
8     function set(uint v) {  
9         value = v;  
10    }  
11    function get() constant returns (uint) {  
12       return value;  
13    }  
14 }  
15 }

Mix

New block created

http://localhost:60357/index.html

Auto reload on save

Sample Ratings

Store:  Save

Query:

Default\_1

GENESIS BLOCK Edit Starting Parameters

BLOCK 1

0x38f388fa... (user1)

→

Sample.Sample()

PENDING TRANSACTIONS

User Account

0x06400992be45bc64a52b5c55d3df84596d6cb4a1 (user2) = 1 Mether

0x38f388fad4a6a35c61c3f88194ec5ae162c8944 (user1) = 1 Mether (is mini...

0xb0dcdc575ef06dc30aaa069d8043c9d463c931c (user4) = 1 Mether

0xfa4b795b491cc1975e89f3c78972c3e2e827c882 (user3) = 1 Mether

Contract Account

Filter by Contract Name or Addresses

Sample - 1025d81198b72fba80a1d4dddad12eeb8360d828 - 0 wei

{  
  value: "12"  
}



# Online Compiler

[illegible]



# ethereum

# Ethereum Studio

Cloud9

File

Edit

Find

View

Goto

Run

Tools

Window

Support

Stop Sandbox

Transactions (2)

Send Contracts to Net

Collaborate

Outline

Debugger

Ethereum Sandbox

Workspace

Navigate

Commands

Test

Changes

workspace

example-project

\_pre

contracts

contract.sol

std.sol

test

web

ethereum.json

gulpfile.js

package.json

README.md

Welcome

contract.sol

1 import "std.sol";

2

3 contract Contract is named("Contract") {

4 function test(bytes32 str) returns (uint256) {

5 return now;

6 }

7 }

Sandbox ID: b9ec8a2f26

0x084f6a99003dae6d3906664fdbf43dd09930d0e3 [NameReg](#)

• Nonce: 0

• Balance: 1234567890123345

• Storage:

uint 0 0x2ad[...]9ba address

0x2f2[...]013 0xded[...]392 address

3db[...]1d4 0x2ad[...]9ba address

604[...]2f1 0x084[...]0e3 address

b6f[...]359 0x179[...]a39 address

c59[...]626 Contract string

f3d[...]a23 NameReg string

.]056

1844bc25aedb27e69bc11b5bda39 [Contract](#)

0

.]056

baalfe0e6bc666dac8fc2697ff9ba [miner]

10010669800000000000

13cd947ec05abc7fe734df8dd826

430

2.2300745198530623e+43

2097153 uint

200010 uint

5b94a16f236a6890cf9e0b1e30392

65

1e+54

[ask us anything](#)

Contract

Contract <ABI>

nameRegAddress

Call

test

str : bytes32 "hello"

Call

ret: Returned value:

0x0057445fbc

named

name : bytes32

bash - "53f66d9d" ×

Immediate

Sandbox Event (NameReg.Register): "

"0x436f6e74726163740000000000000000



# ethereum Solidity

Developer writes contract with functions

Compiler generates init code and dispatcher

At deployment the contract constructor is executed

```
contract Sample
{
    uint value;
    function Sample(uint v) {
        value = v;
    }
    function set(uint v) {
        value = v;
    }
    function get() constant returns (uint) {
        return value;
    }
}
```



# ethereum Types

“Standard” types:

Bool

Int: Signed 256 bit Integer (other sizes available)

UInt: Unsigned 256 bit Integer (other sizes available)

Array: Static and Dynamic

String

Enum



# ethereum Types

Special types:

Address: 160 bit for ethereum address

Fields: balance

Functions: send, call, callcode, delegatecall

Mapping (hashtable-like):

maps from one solidity type to another

contains all keys at construction

Contract Types:

Inherits from address

Have contract-specific functions

```
contract Sample
{
    address a = 0x1239028420398492304f23947289deee24839589;
    mapping (address => uint) balances;
    Sample otherContract;
}
```



ethereum

# Control Flow

If

```
function hello (uint x) returns (uint) {  
    if (x > 5) {  
        return 9;  
    } else {  
        return 10;  
    }  
}
```



ethereum

# Control Flow

For

```
for(uint a = 0; a < 99; a++) {  
  
}
```

While / Break

```
while(true) {  
    break;  
}
```





ethereum

# Other

`this.balance`: gets the balance of the contract

`this.function()`: calls a function by transaction

`super`: inheritance

Automatic getter generation, declare a variable as public

Special variables for blockchain interaction



ethereum

# Global Vars

msg.

sender

value

gas

tx.

origin

gasprice



ethereum

# Global Vars

block.

coinbase

difficulty

timestamp

blockhash

number

Special cryptographic functions (e.g. sha3)



Events for writing to the log

```
Funder[] public funders;  
event FundTransfer(address backer, uint amount, bool isContribution);
```

Import

Standard contracts

Contract inheritance

Code from ancestor copied into child

Still only one contract



# Exercise #1

## **Trusted data feed**

Contains only one field

Can only be changed by the creator

Change Event

Field can be read by other contracts

relevant globals: `msg.sender`



ethereum

# Modifier

Modifiers for code reuse

```
modifier afterDeadline() { if (now >= deadline) _ }
```

*/\* checks if the goal or time limit has been reached and e*

```
function checkGoalReached() afterDeadline {  
    if (amountRaised >= fundingGoal){  
        // sends amountRaised wei to beneficiary account  
        beneficiary.send(amountRaised);  
        FundTransfer(beneficiary, amountRaised, false);  
    }  
}
```



ethereum

# Exceptions

throw: creates and exception

execution aborts, state reverts

cannot be caught on contract functions

all gas is used

```
/* get the offer from the array */  
var offer = offers[id];  
/* throw if offer is not taken */  
if(offer.status != Status.TAKEN) throw;  
/* throw if sender is not the taker */  
if(msg.sender != offer.taker) throw;
```



ethereum

# Sending Ether

Every address or contract object  
has a send method, takes the amount in wei

```
address recipient = 0x1223;  
uint amount = 50 ether;  
recipient.send(amount);
```





# Exercise #1.5

## Trusted data feed

Contains only one field

Can only be changed by the creator

Change Event

Field can be read by other contracts **(for a fee)**

relevant globals: msg.value, throw



# Exercise #2

## Subscription Contract

Manages **one** subscription

Recipient: can withdraw PRICE wei per TIME

Creator: can cancel if there are not outstanding payments

relevant:

`address.send(value)`: send value wei to address

`block.timestamp`: unix timestamp (in seconds)



ethereum

# Contract Calls

Coerce address into contract type

Call the function on that

.value() to send wei

.gas() to limit gas

```
uint public fundingGoal; uint  
token public tokenReward;  
Funder[] public funders;
```

```
// Coerce an address into a contract type  
tokenReward = token(_reward);
```

```
amountRewardEscal = amount;
```

```
// sends a sendCoin message to the tokenReward contract  
tokenReward.sendCoin(msg.sender, amount / price);  
FundTransfer(msg.sender, amount, true);
```

```
// sends a sendCoin message to the tokenReward contract  
tokenReward.sendCoin.value(10 ether)(msg.sender, amount / price);
```



ethereum

# Structs

```
/* data structure to hold information about campaign contributors */  
struct Funder {  
    address addr;  
    uint amount;  
}
```

```
addElementValueOnce(arr, i, Funder({addr: msg.sender, amount: amount}))
```

```
funder.addr.send(funders[i].amount);  
FundTransfer(funder.addr, funder.amount, false);
```



# ethereum Arrays

```
Funder[] public funders;
```

push: adds a new element to the array

```
// push an additional value onto the array  
funders.push(Funder({addr: msg.sender, amount: amount}));
```

```
var funder = funders[i];
```



ethereum

# Solidity

functions can have multiple return values

retrieve values by deconstruction



ethereum

# Visibility

External

Can only be called by a message

Public

Can be called by anyone

Private

Can only be called by the contract itself

Internal

Cannot be called by a message



ethereum

# Enums

```
/* Status enum for the 3 possible states */  
enum Status { OFFERED, TAKEN, CONFIRMED}
```

```
/* throw if the offer has already been taken  
if(offer.status != Status.OFFERED) throw;
```





# Exercise #3

## **Market Contract**

Seller can add offers (with name and price)

Buyer can take offers (by sending the right amount)

Buyer can confirm the offer (and release funds)