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

ethereum

Fundamentals

Message

- Sender
- Recipient
- Value (can be 0)
- Data
- Return Value
- Gaslimit

Executes either completely or not at all

ethereum

Fundamentals

Contract

- 160 bit address
- Balance
- EVM Bytecode
- Runs at every received message
- Has a persisten 256-to-256 bit storage
  - Private
  - Expensive
- Can spawn new messages

ethereum

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

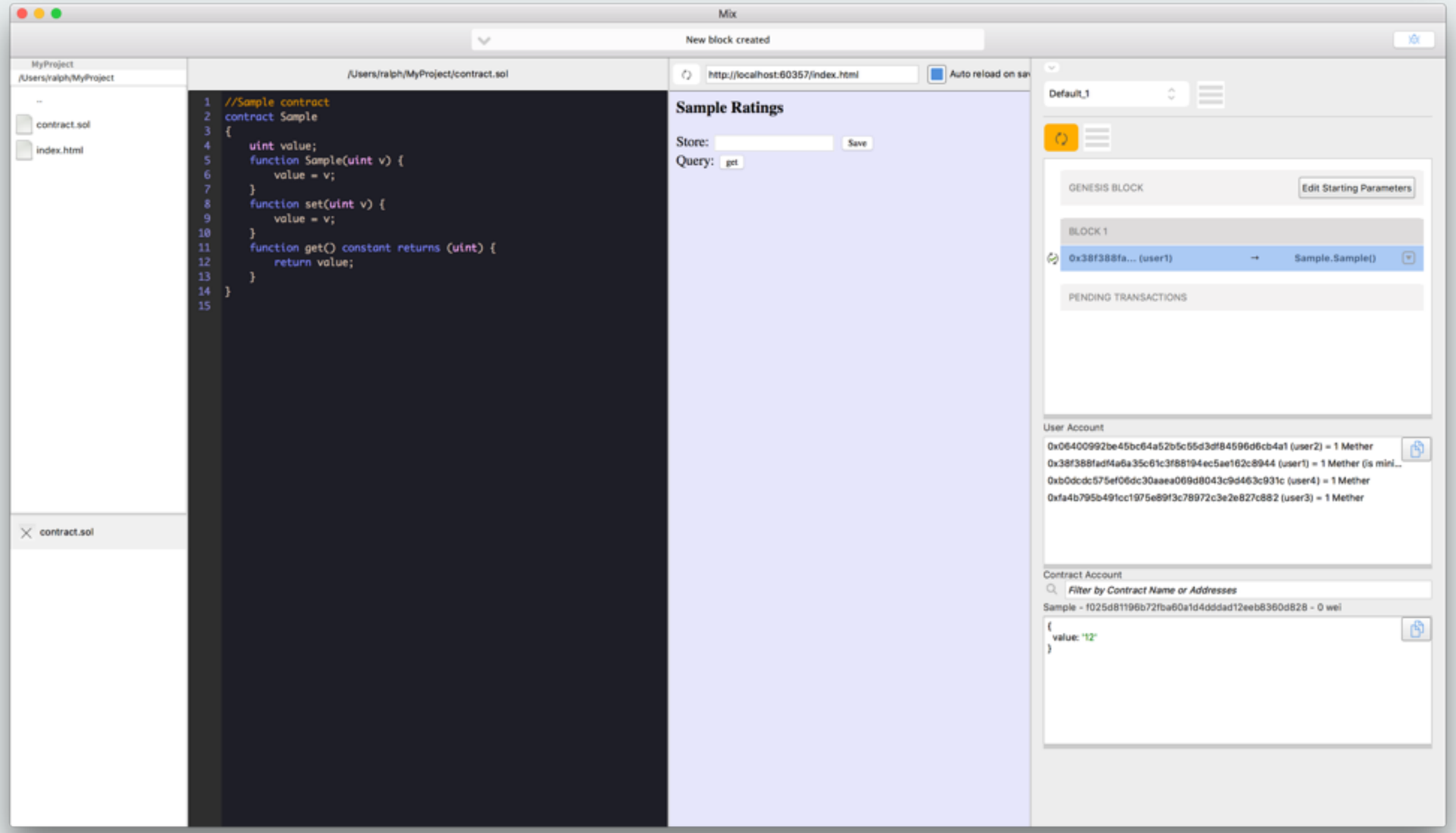
Light Clients

Logging

Self Destruct / Suicide

ethereum

Mix IDE



ethereum Online Compiler

Untitled

```
contract Subscription {

}
```

0xca35b7d915458ef54 Transaction origin

10 ether Value (e.g. .7 ether or 5 wei, defaults to ether)

Subscription 26 bytes

At Address | Create

Bytecode 6060604052600a8060106000396000f3606060405260008565b00

Interface []

Web3 deploy var subscriptionContract = web3.eth.contract([]);

uDApp [{"name":"Subscription","interface":"
[]\n","bytecode":"6060604052600a8060106000396000f3606060405260008565b0

Toggle Details

Solidity Interface: contract Subscription{}

Opcodes: PUSH1 0x60 PUSH1 0x40 MSTORE PUSH1 0xA DUP1 PUSH1 0x10 PUSH1 0x0 CODEC

Functions

Gas Estimates

```
Creation: 39 + 2000
External:
Internal:
```

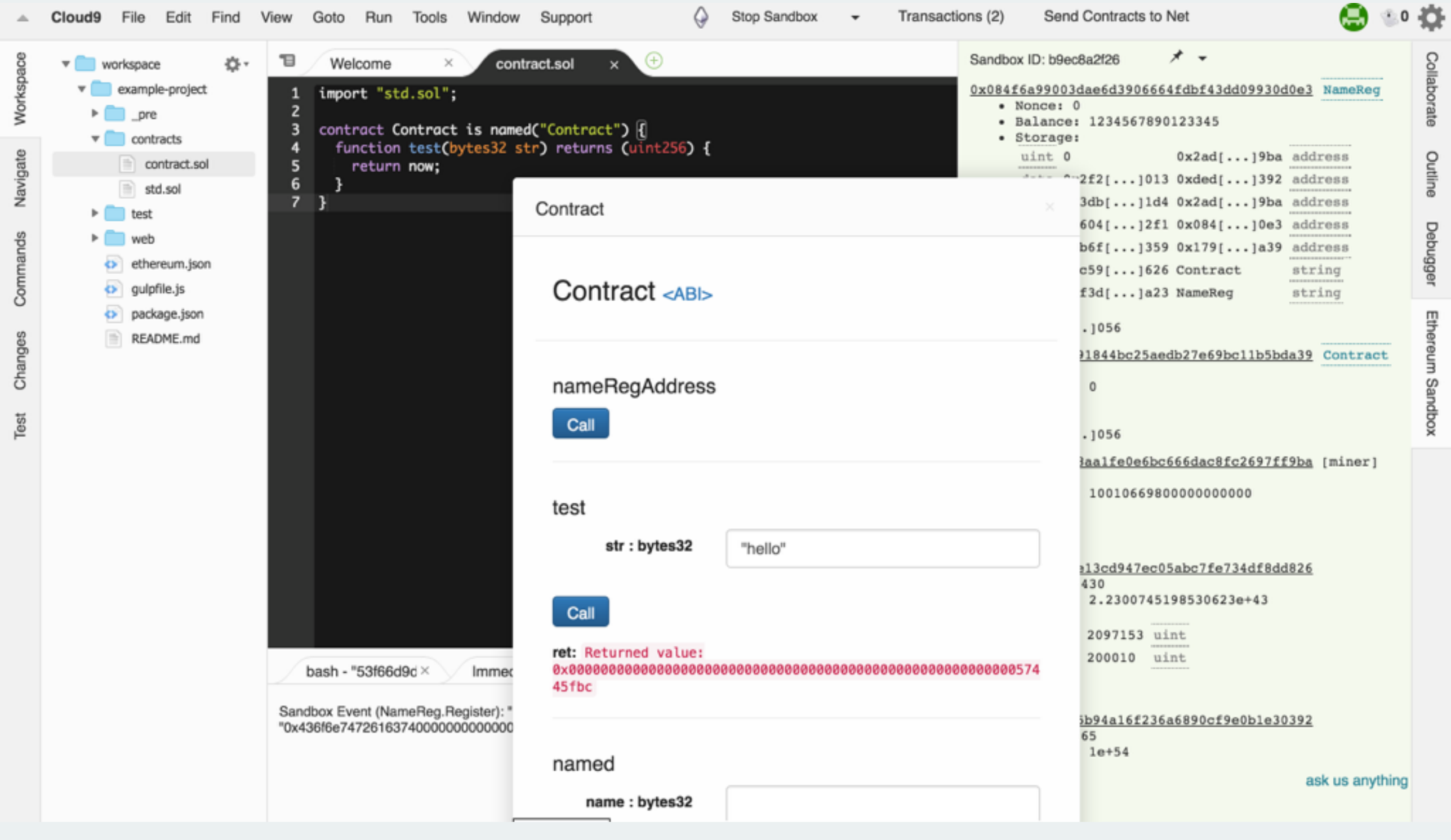
Runtime Bytecode: 6060604052600a8060106000396000f3606060405260008565b00

Assembly

```
.code
  PUSH 60                  contract Subscription {\n
...
  PUSH 40                  contract Subscription {\n
...
  MSTORE                   contract Subscription {\n
...
  PUSH #[$] 0000000000000000000000000000000000000000000000000000000000000000
```

New File

ethereum Ethereum Studio



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;
    }
}
```

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;
}
```

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;
}
```

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

Global Vars

msg.

- sender
- value
- gas

tx.

- origin
- gasprice

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

```
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;
```

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);
```

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

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

Structs

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

```
n(Funder({addr: msg.sender, amount: amount}))
```

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

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

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)