

CS 252:

*Advanced Programming Language Principles*



# Ethereum and Solidity

Prof. Tom Austin

San José State University

# Bitcoin (BTC)



*bitcoin*

- Protocol designed by Satoshi Nakamoto in 2008  
<https://bitcoin.org/bitcoin.pdf>
- First Bitcoin client launched in 2009
- Peer-to-peer – no centralized control
  - Every client keeps track of the history of all bitcoins

# Bitcoin "Script" Language

- Forth-like
  - Reverse-Polish notation
  - Stack based
- Lock: `scriptPubKey`
- Unlock: `scriptSig`
- <https://en.bitcoin.it/wiki/Script>

# Sample Transaction Output

```
"vout": [  
  { "value": 0.01500000,  
    "scriptPubKey": "OP_DUP OP_HASH160  
      ab6802... OP_EQUALVERIFY  
      OP_CHECKSIG" },  
  { "value": 0.08450000,  
    "scriptPubKey": "OP_DUP OP_HASH160  
      7f9b1a... OP_EQUALVERIFY  
      OP_CHECKSIG" },  
]
```

# Sample Script

```
2 7 OP_ADD 3 OP_SUB 1 OP_ADD 7 OP_EQUAL
```

(in-class)

# Sample Script

## Pay-to-public-key-hash (P2PKH)

```
<sig> <PubK> DUP HASH160 <PubKHash>  
EQUALVERIFY CHECKSIG
```

(in-class)

# Script Limitations

- No loops.
- No complex control flow.
- Not Turing complete.
- No division.

# Ethereum

- Smart contracts for building decentralized applications (dApps).
- Almost Turing complete.
  - "Gas" to pay for computation.



# Some Brief Ethereum Facts

- Number 2 cryptocurrency by market cap.
- Core developers
  - Vitalik Buterin (creator)
  - Gavin Wood
- Block 0 mined July 30, 2015



# Ethereum Prehistory: Mastercoin

- Protocol layer on top of Bitcoin
- Focused on financial contracts
  - Two-party contracts with enforced terms
- October 2013: Vitalik suggested a more flexible scripting language
  - More limited vision
  - Not Turing-complete

# December 2013: Ethereum Proposal

- The name "Ethereum" first appears in print
- Transaction fees for different actions included
  - Computational steps paid for in ether (this concept changed later)
  - Once fees exhausted, processing stops
- *Contracts* became accounts in their own right

# More History

- Gavin Wood joins project
  - Designs the Ethereum Virtual Machine (EVM).
  - Writes "The Yellow Paper" in 2014.
- Gas model changes
  - Miners explicitly vote on gas price.
  - Previous approaches allowed them to implicitly do this anyway.

# Account Types

- *Externally owned accounts* (EOAs)
  - Have a private key
- Contract accounts
  - Have contract code
  - No private key
    - Cannot initiate transactions
  - Can react to transactions and call other contracts
  - Contain data

# Common Features with Bitcoin

- Digital currency
  - Called *ether* (ETH)
    - Not "ethereum"
  - Smallest unit: *wei*
- Proof-of-work blockchain
  - Ethash – designed to be ASIC-resistant
  - Much quicker: 14-15 second block time
  - Plans to move to *proof-of-stake* (Casper)
- Peer-to-peer network

# Differences from Bitcoin

- Quasi-Turing complete virtual machine
  - Brings up a lot of security issues
- Gas
  - Prevents denial-of-service attacks
  - Transactions specify:
    - ETH earmarked for gas
    - gas-rate

# Decentralized Applications (DApps)

- Also referred to as dApps, Dapps, and ÐApps
  - Ð is the Old-English letter 'Eth'
- Written in a smart contract language
  - Solidity is the most prevalent

# Smart Contracts

# Smart Contracts

(Definition from *Mastering Ethereum*)

*Immutable computer programs that run deterministically in the context of an Ethereum Virtual Machine as part of the Ethereum network protocol—i.e., on the decentralized world computer.*

# Smart Contract Life Cycle

1. Published to the *zero address*.
  - `0x00`
  - Author has no special rights to a contract, unless the contract is written that way.
2. Invoked by transaction.
3. May be destroyed.
  - Only if creator configured it that way.

# High-level Languages for EVM

- LLL – Lisp-like language.
  - Oldest, but rarely used.
- Serpent
  - Python-ish
- Solidity
  - JavaScript-ish
- Vyper
  - Also Python-ish
- Bamboo
  - Erlang-ish

# High-level Languages for EVM

- LLL – Lisp-like language.
  - Oldest, but rarely used.
- Serpent
  - Python-ish
- **Solidity**
  - JavaScript-ish
- Vyper
  - Also Python-ish
- Bamboo
  - Erlang-ish

# Solidity

- Created by Gavin Wood.
- Most popular HLL for Ethereum today.

# Solidity Data Types

(not exhaustive)

- `bool`
- `int, uint`
  - Variants in 8, 16, 32, ..., 256
  - Default is 256
- `fixed, ufixed`
- `address`
- **Arrays**
- **Time units**
- **Ether units: `wei, finney, szabo, and ether`**

# Global Variables

- `msg` – the transaction call.
  - Fields: `sender`, `value`, `gas`, `data`, `sig`
- `tx` – the transaction.
  - Fields: `gasprice`
- `block` – the block the transaction is in.
  - Fields: `coinbase`, `difficulty`, `gaslimit`, `number`, `timestamp` (in seconds since epoch)

# Constructing and Destroying Contracts

- Created with `constructor`.
  - Older versions used contract name
- Destroyed with `selfdestruct`.
  - Person who destroys it claims the contract's ether.
  - Only if enabled by author.

# Function Syntax

```
function FunctionName  
  ([parameters]) {public|  
private|internal|external}  
  [pure|constant|view|payable]  
  [modifiers]  
  [returns (return types)]
```

# Function Modifiers

- Functions that modify other functions
- Use an underscore (`_`) as a placeholder for the modified function

```
modifier onlyOwner {  
    require (msg.sender == owner);  
    _;  
}
```

# Function Restricting Access

```
function takeFunds(amt) public {  
    require (msg.sender == owner);  
    msg.sender.transfer(amt);  
}
```

# Using Function Modifier

```
function takeFunds(amt) public onlyOwner {  
    msg.sender.transfer(amt);  
}
```

# Error handling

- Guarantee state.
  - Throw an exception if false.
- `assert`
  - Used only to catch internal programming errors
- `require`
  - Used to validate external input
  - May be given 2<sup>nd</sup> argument for better error handling

# Review Test Faucet Contract

(in-class)

# Suggested Reading

- The Beige Paper
  - <https://github.com/chronaeon/beigepaper/blob/master/beigepaper.pdf>
  - Less formal version of the Yellow Paper (<https://ethereum.github.io/yellowpaper/paper.pdf>)
- *Mastering Ethereum*, by Andreas M. Antonopoulos and Gavin Wood. <https://github.com/ethereumbook/ethereumbook>

# Lab: Distributed Lottery in Ethereum

There can be exactly 3 players.

Each player contributes ether and pick a random number.

Once the last player has contributed, the winner's address is selected.

The winner can then destroy the smart contract to claim the winnings.

More details in Canvas.