



# Ethereum Workshop

An Introduction to Tools, Solidity & Smart  
Contracts

# Preparation

- Download and install:
  - Geth – an Ethereum client  
Installation instructions:  
<https://github.com/ethereum/go-ethereum/wiki/Building-Ethereum>
  - Mist – a DApp browser

Download the file starting with Mist-\* (you need to scroll down a bit to find the list of files)

<https://github.com/ethereum/mist/releases>



# Agenda

- 1) A brief introduction to Ethereum
- 2) Setting up a private blockchain
- 3) Interacting with the blockchain
- 4) Mist
- 5) Solidity & Smart Contracts
- 6) Remix IDE



# **1. A brief introduction to Ethereum**



# Blockchain Introduction

- TODO: Copy from older presentations



# Blockchain Introduction

- TODO: Copy from older presentations



# Blockchain Introduction

- TODO: Copy from older presentations



## **2. Setting up a private blockchain**



# Accounts

```
> geth --datadir ~/.ethereum/workshop  
account new
```

- `geth` – **Go Ethereum** client
- `--datadir <DIRECTORY>` – Store all data (incl. the blockchain in DIRECTORY)
- `account new` – Create a new account

# Windows!



- On Windows machines, replace:  
`~/ .ethereum/workshop`  
with:
- `C:\Users\[USERNAME]\workshop`

# Accounts

```
> geth --datadir ~/.ethereum/workshop  
account list
```

- `account list` – List all existing accounts

# The genesis block

```
{  
  "config": {  
    "chainId": 42,  
    "homesteadBlock": 0,  
    "eip155Block": 0,  
    "eip158Block": 0  
  },  
  "nonce": "0x000000000000000042",  
  "timestamp": "0x00",  
  "parentHash": "0x0000000000[... ]000000000000",  
  "extraData": "0x00",  
  "gasLimit": "0x4c4b40",  
  "difficulty": "0x6666",  
  "mixhash": "0x0000000000000000[... ]000000000000",  
  "coinbase": "0x000000000000000000000000000000000000000000000000"  
}
```

# Initialize the blockchain

```
> geth --datadir ~/.ethereum/workshop  
init genesis.json
```

- `init <GENESIS FILE>` – Initialize a new blockchain from a genesis file

# Start a miner

```
> geth --datadir ~/.ethereum/workshop  
--networkid 42 --mine --minerthreads  
1
```

- `--networkid <NUMBER>` – A unique number for this blockchain network
- `--mine` – Make this blockchain node a miner
- `--minerthreads <NUMBER>` – How many threads (which indirectly means CPUs) are used for mining



# Predefined network ids

- 1) Frontier
- 2) Morden (disused)
- 3) Ropsten
- 4) Rinkeby

# Plan B if something doesn't work

```
> geth --dev
```

- `--dev` – Developer mode: pre-configured private network – Cannot connect to other nodes



# Start a console

```
> geth attach  
ipc:///home/jonas/.ethereum/workshop/g  
eth.ipc
```

- `attach <PATH>` – Attach a console to a running geth instance using IPC

# Windows!



- On Windows machines run just:  
`> geth attach`

# Windows!



- On Windows machines run just:  
`> geth attach`



## **2. Interacting with the blockchain**

# admin API

- `admin.nodeInfo` – Gives us the enode id and a bunch of useful information
- `admin.peers` – Lists all connected nodes our node knows
- `admin.addPeer("enode://fc[...]03")` – Manually add another node

<https://github.com/ethereum/go-ethereum/wiki/JavaScript-Console#management-api-reference>



# personal API

- `personal.listAccounts` – List of all (local) accounts
- `personal.unlockAccount("0xc73[...]5b")` – Lists all connected nodes our node knows

<https://github.com/ethereum/go-ethereum/wiki/JavaScript-Console#management-api-reference>



# web3js

- `web3.eth.getBalance("0xc[...]5b")` – Get balance of account. This works for all accounts.

<https://github.com/ethereum/wiki/wiki/JavaScript-API#web3js-api-reference>

# Dealing with numbers

- `number.toString(10)` – Converts a bignumber to a human-readable string  
<https://github.com/ethereum/wiki/wiki/JavaScript-API#a-note-on-big-numbers-in-web3js>  
<https://github.com/MikeMcl/bignumber.js/>
- `web3.fromWei(number, "ether")` – Converts from wei to ether  
<https://github.com/ethereum/wiki/wiki/JavaScript-API#web3fromwei>  
<https://converter.murkin.me/>
- `web3.toWei(number, "ether")` – From ether to wei



# web3js

- ```
web3.eth.sendTransaction(  
  {"from": "0xc73e[...]2cfbc025b",  
   "to": "0x00[...]00",  
   "value": 111111  
})
```

  - Send wei from an address to another address



## **3. Mist**

# Starting Mist

```
> mist --rpc  
/home/jonas/.ethereum/workshop/geth.ip  
c
```

- `--rpc` – Path to node IPC socket file OR HTTP RPC hostport

# Windows!



- On Windows machines run:  

```
> "C:\Program Files\Mist\Mist.exe"  
--rpc \\.pipe\geth.ipc
```

# Mist

The screenshot displays the Mist Ethereum wallet application. The top menu bar includes 'Mist', 'File', 'Edit', 'View', 'Develop', 'Window', and 'Help'. The main interface is titled 'Ethereum Wallet' and shows a URL bar with a transaction link. On the right, the 'BALANCE' is 58,396.00 ETH. The 'SEND' tab is active, showing a 'Send funds' screen. The 'FROM' field is 'Main account (Etherbase) - 306.00 ETH'. The 'TO' field is a hexadecimal address. The 'AMOUNT' field is '123'. Below it, a checkbox for 'Send everything' is unchecked. A summary line states 'You want to send 123 ETH'. A blue button 'SHOW MORE OPTIONS' is present. The 'SELECT FEE' section shows a fee of 0.000378018 ETH. A sidebar on the left shows network status: 44.4 KH/s, 11,879 1, and 14s. A 'Private' status indicator is at the bottom left. A disclaimer at the bottom right states: 'This is the most amount of money that might be used to process this transaction. Your transaction will be mined ~~probably~~ within 30 seconds'.

Mist File Edit View Develop Window Help

Ethereum Wallet <https://wallet.ethereum.org> send-from ► 0xC73E37361DE2D45e635E433d4F0d66e2CFbC025b

WALLETS SEND CONTRACTS BALANCE 58,396.00 ETH\*

## Send funds

FROM TO

Main account (Etherbase) - 306.00 ETH 0xee3099424b2e9761b42364efe960a60f690519e3

AMOUNT

123

☐ Send everything

You want to send **123 ETH**.

SHOW MORE OPTIONS

SELECT FEE

0.000378018 ETH

44.4 KH/s  
11,879 1  
14s  
Private

ETHER 305.9999999999977778 ETH

This is the most amount of money that might be used to process this transaction.  
Your transaction will be mined ~~probably~~ within 30 seconds



## **4. Solidity & Smart Contracts**

# Smart Contracts

```
pragma solidity ^0.4.15;
```

```
contract MyCoin {
```

```
    mapping (address => uint) balances;
```

```
    function MyCoin() {
```

```
        balances[tx.origin] = 10000;
```

```
    }
```

```
    function sendCoin(address receiver, uint amount) returns(bool sufficient) {
```

```
        if (balances[msg.sender] < amount) return false;
```

```
        balances[msg.sender] -= amount;
```

```
        balances[receiver] += amount;
```

```
        return true;
```

```
    }
```

```
    function getBalance(address addr) returns(uint) {
```

```
        return balances[addr];
```

```
    }
```

```
}
```

# Smart Contracts

```
pragma solidity ^0.4.15;
```

```
contract MyCoin {  
    mapping (address => uint) balances;
```

```
    function MyCoin() {  
        balances[tx.origin] = 10000;  
    }
```

```
    function sendCoin(address receiver, uint amount) returns(bool sufficient) {  
        if (balances[msg.sender] < amount) return false;  
        balances[msg.sender] -= amount;  
        balances[receiver] += amount;  
        return true;  
    }
```

```
    function getBalance(address addr) returns(uint) {  
        return balances[addr];  
    }  
}
```



# Smart Contracts

```
pragma solidity ^0.4.15;
```

```
contract MyCoin {  
    mapping (address => uint) balances;
```

```
    function MyCoin() {  
        balances[tx.origin] = 10000;  
    }
```

```
    function sendCoin(address receiver, uint amount) returns(bool sufficient) {  
        if (balances[msg.sender] < amount) return false;  
        balances[msg.sender] -= amount;  
        balances[receiver] += amount;  
        return true;  
    }
```

```
    function getBalance(address addr) returns(uint) {  
        return balances[addr];  
    }  
}
```

# Smart Contracts

```
pragma solidity ^0.4.15;
```

```
contract MyCoin {  
    mapping (address => uint) balances;
```

```
    function MyCoin() {  
        balances[tx.origin] = 10000;  
    }
```

```
    function sendCoin(address receiver, uint amount) returns(bool sufficient) {  
        if (balances[msg.sender] < amount) return false;  
        balances[msg.sender] -= amount;  
        balances[receiver] += amount;  
        return true;  
    }
```

```
    function getBalance(address addr) returns(uint) {  
        return balances[addr];  
    }  
}
```

# Smart Contracts

```
pragma solidity ^0.4.15;
```

```
contract MyCoin {  
    mapping (address => uint) balances;
```

```
    function MyCoin() {  
        balances[tx.origin] = 10000;  
    }
```

```
    function sendCoin(address receiver, uint amount) returns(bool sufficient) {  
        if (balances[msg.sender] < amount) return false;  
        balances[msg.sender] -= amount;  
        balances[receiver] += amount;  
        return true;  
    }
```

```
    function getBalance(address addr) returns(uint) {  
        return balances[addr];  
    }
```

```
}
```



## **5. Remix IDE**

# Remix

The screenshot displays the Remix Solidity IDE interface. The main editor shows the source code for a contract named `MyCoin` in `MyCoin.sol`. The code is as follows:

```
1 pragma solidity ^0.4.14;
2
3 contract MyCoin {
4     mapping (address => uint) balances;
5
6     function MyCoin() {
7         balances[tx.origin] = 10000;
8     }
9
10    function sendCoin(address receiver, uint amount) returns(bool sufficient) {
11        if (balances[msg.sender] < amount) return false;
12        balances[msg.sender] -= amount;
13        balances[receiver] += amount;
14        return true;
15    }
16
17    function getBalance(address addr) returns(uint) {
18        return balances[addr];
19    }
20 }
21
```

The right sidebar contains the deployment and analysis tools. The top tabs are `Contract`, `Settings`, `Files`, `Debugger`, and `Analysis`. The `Contract` tab is active, showing the following configuration:

- Environment:** Injected Web3
- Account:** 0xc73...c025b (1415.99999999999977778 €)
- Gas limit:** 3000000
- Value:** 0

Below the configuration, there are buttons for `Publish` (pink), `Attach` (green), `Transact` (red), `Transact(Payable)` (red), and `Call` (blue). A dropdown menu shows the selected contract: `browser/MyCoin.sol:MyCoin`. At the bottom, there are buttons for `Publish`, `At Address`, and `Cre...`. A link for `Contract details (bytecode, interface etc.)` is also present.