FS20

BLOCKCHAIN, BITCOIN, ETHEREUM

Introduction, part 2

T. Bocek

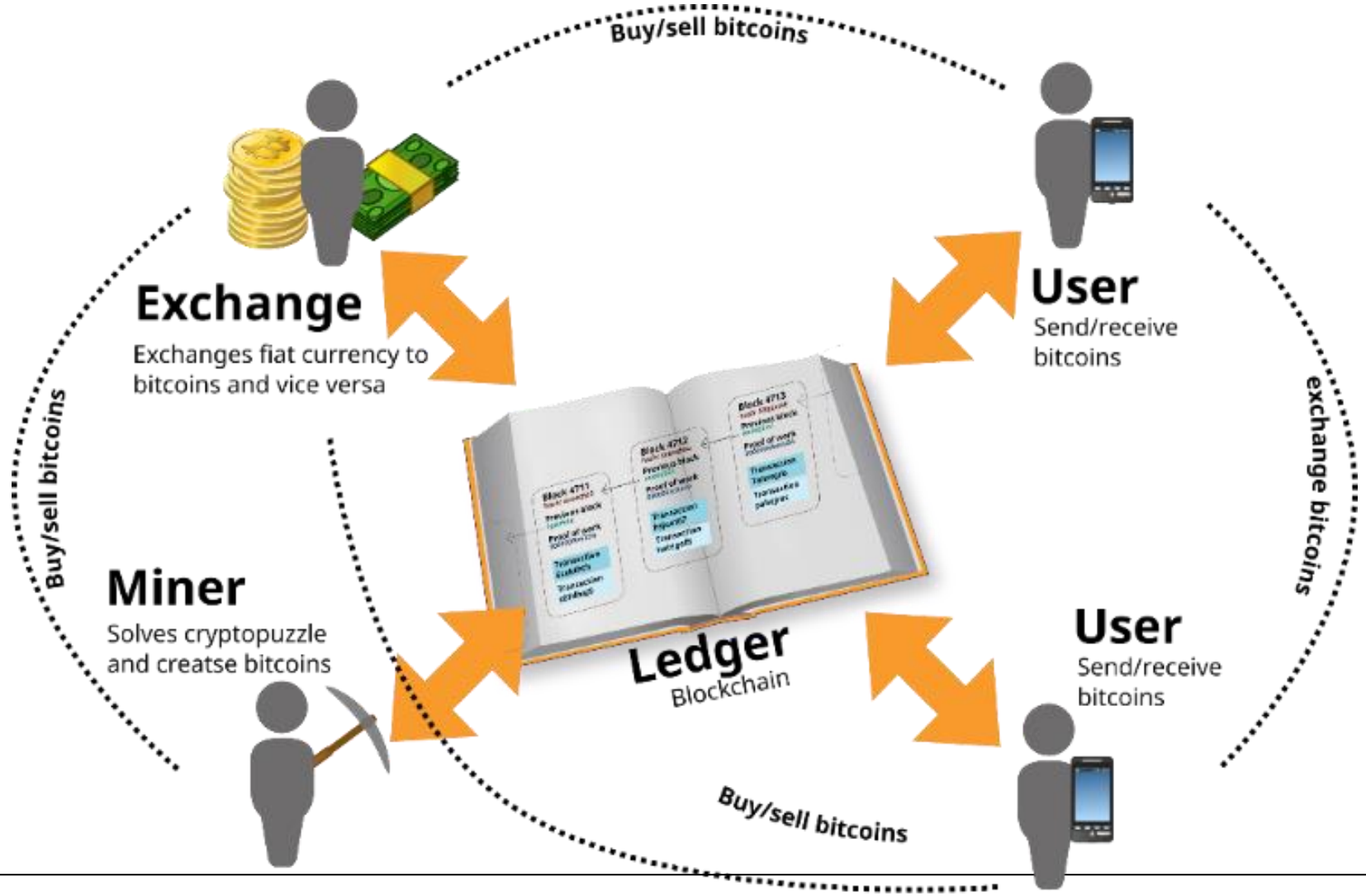
thomas.bocek@hsr.ch

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

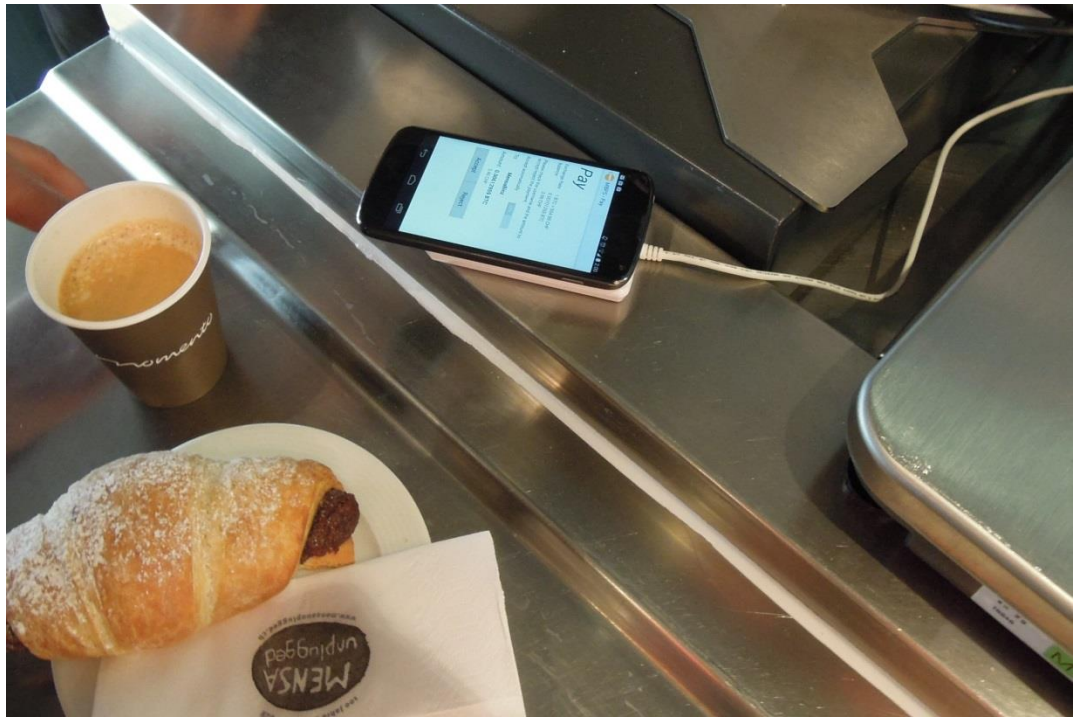
Summary: Bitcoin Stakeholders

- **Building blocks**
  - https://andersbrownworth.com/blockchain/



Bitcoins at UZH: Evaluation: Mensa Test Run

- **Designed and implemented a Bitcoin payment system**
- **One week test run from 10th to 14th of February 2014**
- **In collaboration with the Mensa Binzmühle**
- **Pay all consumptions in Mensa with Bitcoins**
- **Reduce Bitcoin volatility risk by immediate trades on Bitstamp.net**
  - After selling Bitcoins at the exchange point → Buy the same amount of Bitcoins
    - Keep the balance of the exchange point constant
  - After the Mensa receives Bitcoins → Sell these Bitcoins
    - Since the Mensa wants to receive CHF at the end, the equivalent amount is assured in this way



Example payment at Mensa Binzmühle

Bitcoins at UZH: Evaluation: Conclusions

- **NFC handling caused many problems because users are inexperienced with NFC**
- **Android 4.4 restriction → too big entry barrier at the moment / below 4.4 no two-way NFC possible**

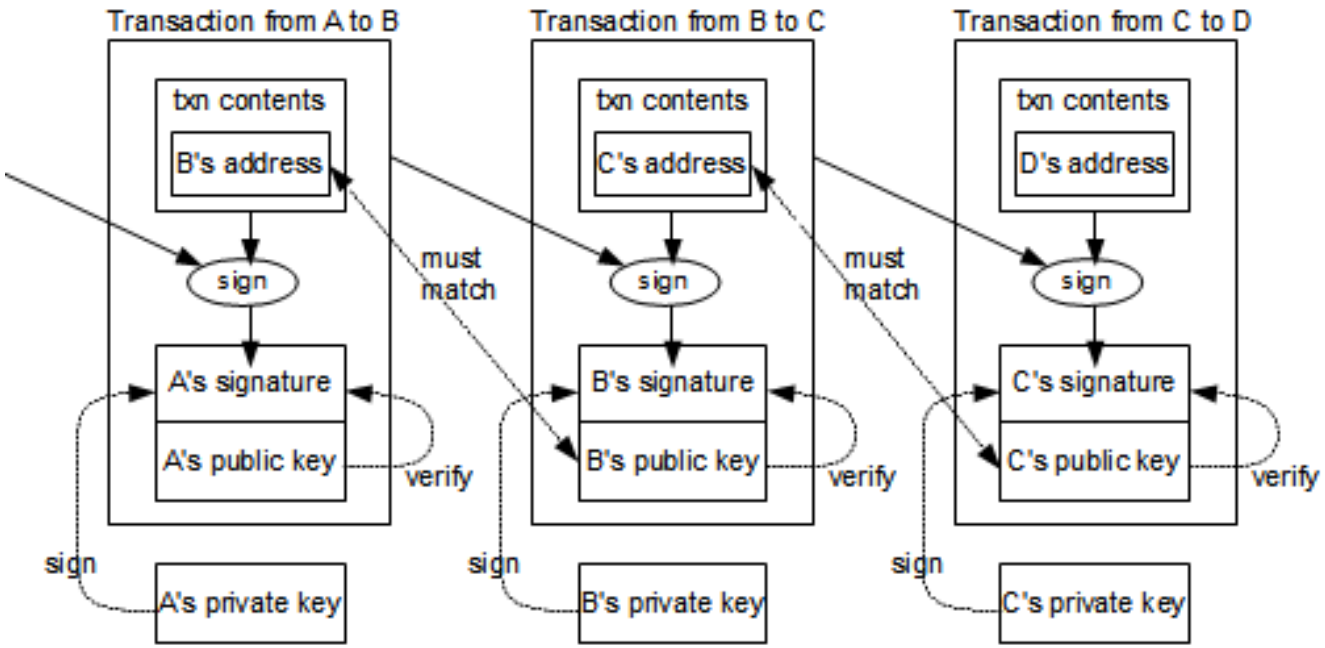
USD/mBTC Exchange Rate Drop – February 10, 2014

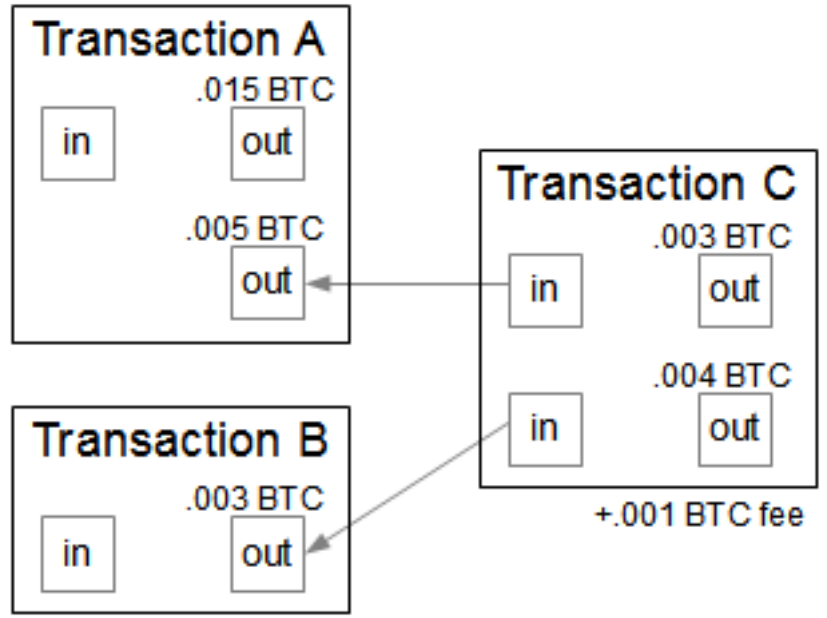
Discussion

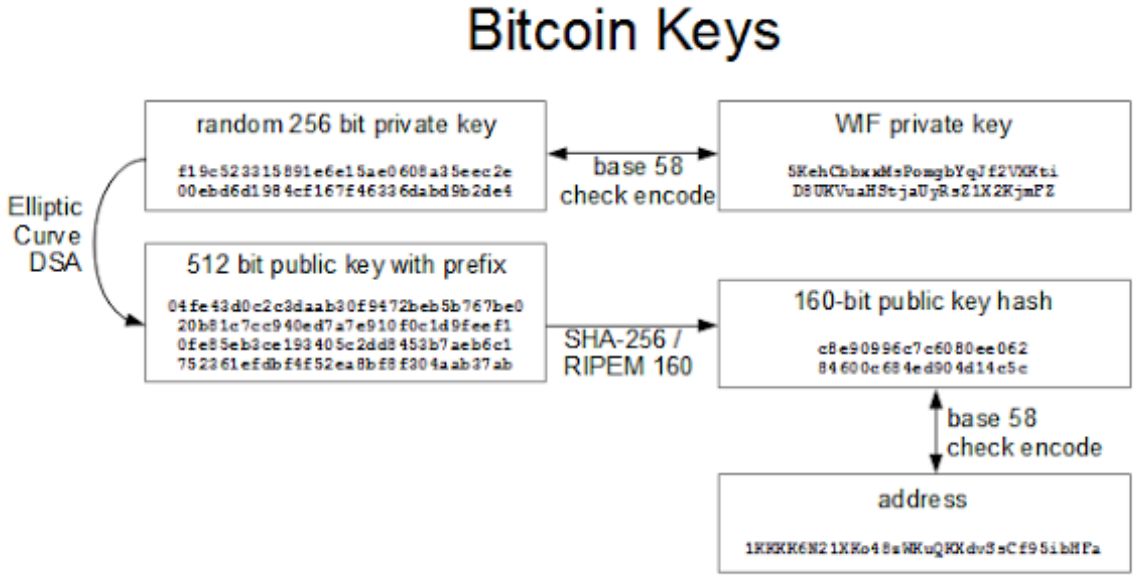
- **But! wrt ASIC – *"It shows that Bitcoin is considered stable enough for a company to invest a LOT of money on design and production on it, which is very good. ASICs were bound to happen if Bitcoin survived long enough."***
- **But! Bitcoin does not intend to replace VISA**
  - Still affordable by (some) individuals (buy every 2.6 weeks a new 1 TB disk storage)
  - Better comparable to wire transfers
- **But! Moore's, Nielson's law, and other laws (lecture 1)**
  - Moore's law: number of transistors doubles every two years
  - Nielson's law: Bandwidth grows 50% per year
  - Kryder's Law: Space per unit cost has doubled roughly every 14 months
- **But! Lightning**
  - TX fees up to 30 USD → micro payments
  - Multisig, hashed timelock → fast and cheap transactions
  - Mulitple efforts to make a blockchain scalable

Bitcoin in Detail

- **Good information:**
  http://www.righto.com/2014/02/bitcoins-hard-way-using-raw-bitcoin.html

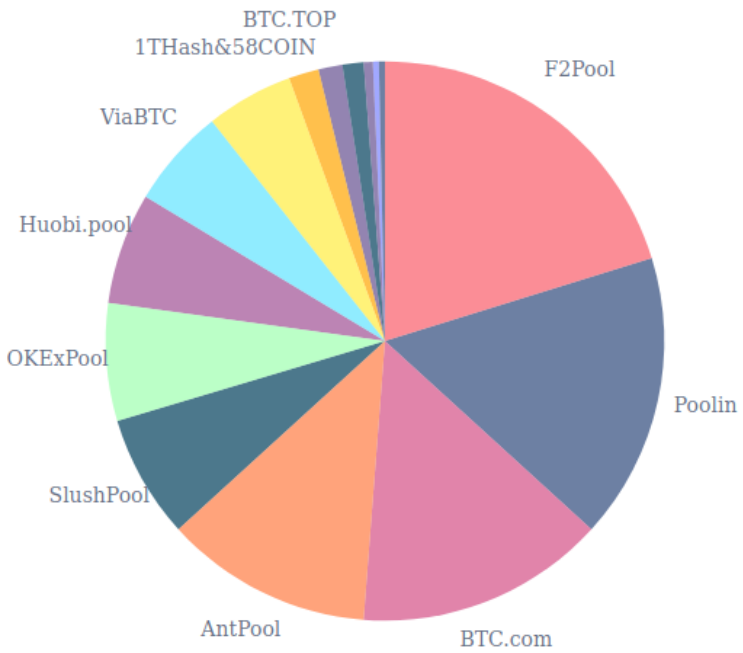






51% Attack

- **Control over 50% of the scarce resources**
  - Pools: cooperative puzzle solving
  - Solo: competitive puzzle solving

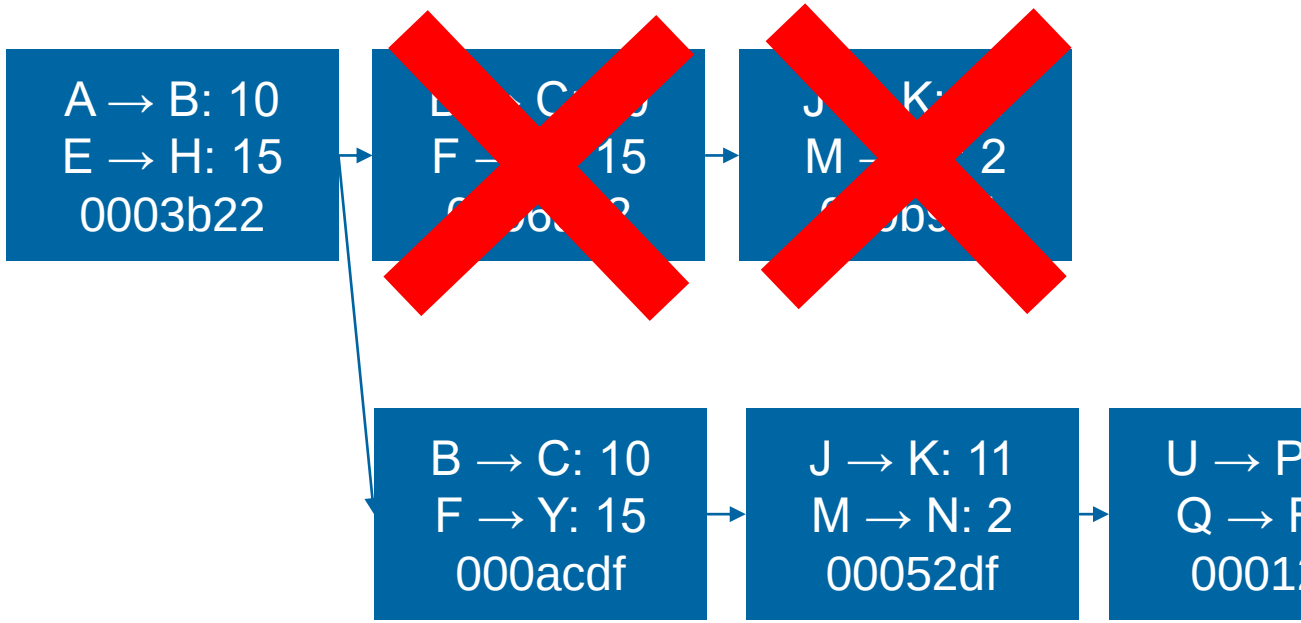


http://blockchain.info/pools

- **03.01.2019:** BTC.TOP, Mining Pool, Controls Over 50% Of The BCH Hashrate
  - Bitcoin Cash
- **25.06.2018:** Bitmain's Mining Pools Now Control Nearly 51 Percent of the Bitcoin Hashrate
  - Was at ~42%, now ~30%
- **07.01.2019:** Deep Chain Reorganization Detected on Ethereum Classic (ETC)
  - "The total value of the double spends that we have observed thus far is 219,500 ETC (~$1.1M)."
- **23.04.2020:** DeFi Platform Suffers 51% Attack From Its Top Miners — or Does It?
  - "resulted in $6.7 million worth of the USD-pegged stablecoin pUSD"

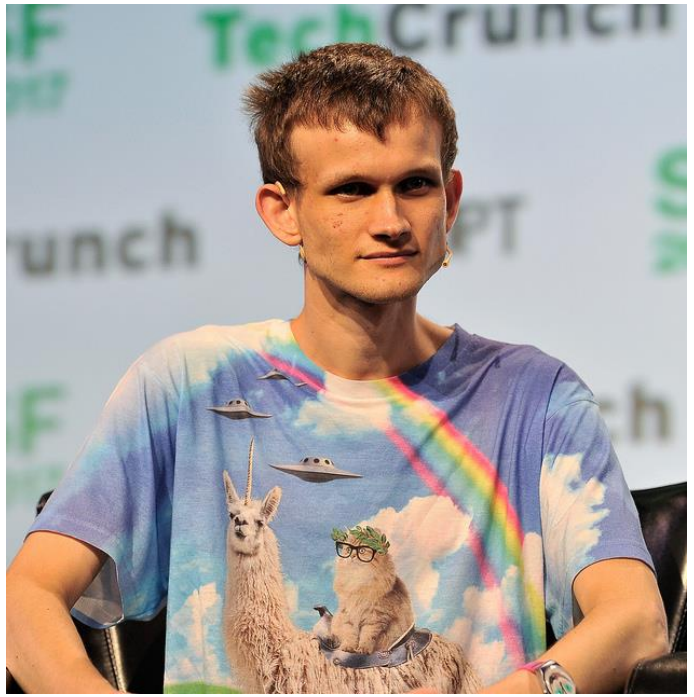
51% Attack

- **“If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains.”**
  - https://bitcoin.org/bitcoin.pdf
- **PoW: majority of hashing power, PoS: majority of coins**
- **How expensive is a 51% attack?**
  - Buy an attack?
- **Double spend, or rollback transactions**
  - X is an exchange
  - Mine secretly, Y is your address
  - X arrived – payout (1 block conf.)
  - You mine faster, broadcast secret chain
  - Tx F→X: 15 never happened, goes to Y



Bitcoin / Ethereum

- **“Issues” with Bitcoin**
  - Slow development, implementing new features difficult
    - Many Bitcoin hardforks (segregated witness vs. increasing block size voting) Cash vs. SV
  - Bitcoin Script limited
    - Lightning network
- **Ethereum (1 ETH ~187$)**
  - Generalized blockchain (loops, arithemitcs, etc.)
  - White paper released in December 2013
  - Protocols designed from scratch (not like Litecoin, Peercoin)
  - Ethereum foundation located in Zug (initiator known) - non-profit foundation
  - Mining reward ~ block every ~14s – ~2 ETH (“always”, unlike Bitcoin)



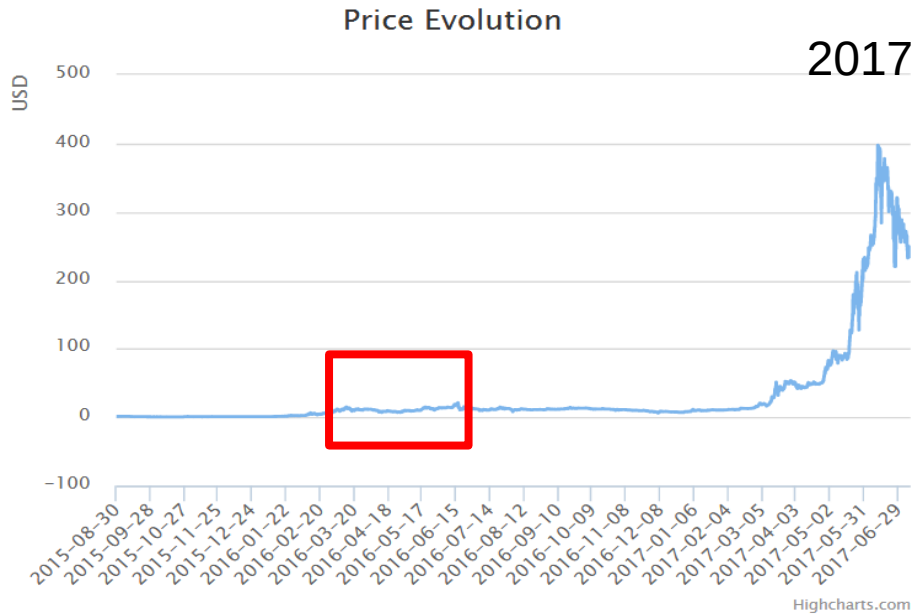
Vitalik Buterin

Ethereum History

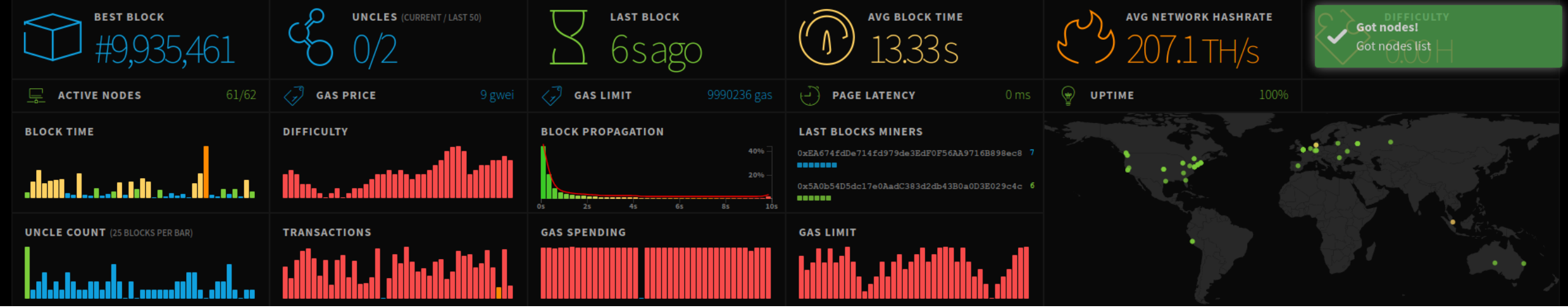
- **Olympic (past) – released 09.05.2015**
  - Last Ethereum Proof-of-Concept series
  - “Olympic will feature a total prize fund of up to 25,000 ether” (now 3.5m USD)
- **Frontier (past) - released 30.07.2015**
  - Main public network, “Beta”/use at your own risk
- **Homestead (past) - released 14.03.2016**
  - Public network considered “stable”, integrate critical protocol changes
- **Metropolis**
  - Byzantium - released 16.10.2017
    - Initial supports for ZKP (private smart contracts), reducing mining reward to 3 ETH
  - Constantinople – released 27.02.2019
    - Bitshifting, mining reward 2 ETH (delay difficulty bomb), other optimizations
  - Istanbul - released 08.12.2019, gas cost changes, DDoS resilience, Zcash interop
  - Muir Glacier- released 01.01.2020 (delay difficulty bomb)
- **Serenity (future)**
  - Scalability, proof-of-stake, Sharding, ZKP

Ethereum Stats

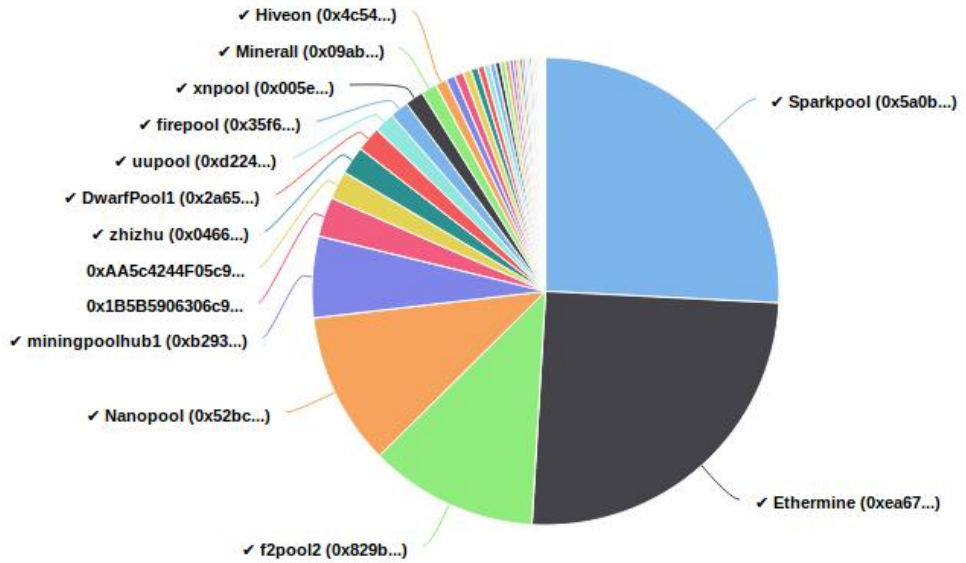
- **Basic Stats**
  - 2nd in market cap ~ 20b USD
  - Transactions (highest ~15.5 TPS) now ~826’000 per day
  - Node count (~7’000 nodes)
  - Blocksize ~25KB
  - Accounts (95mio)

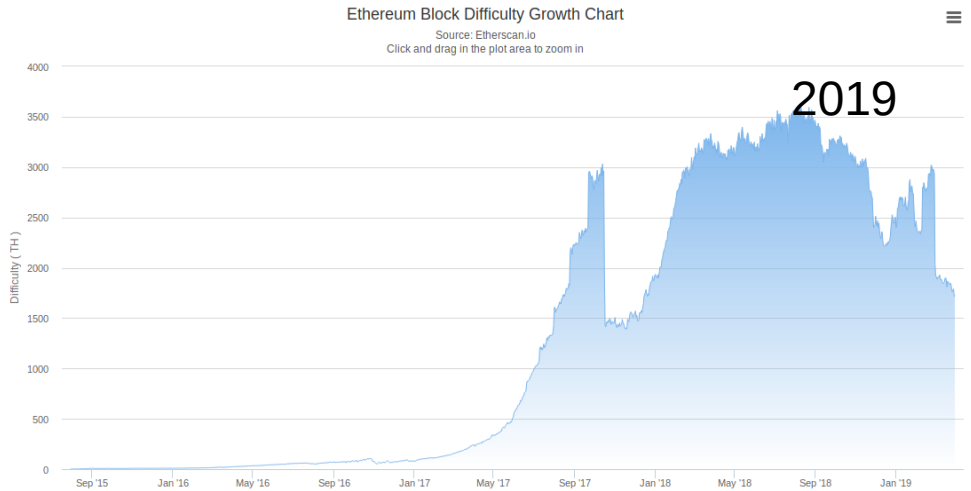


HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz



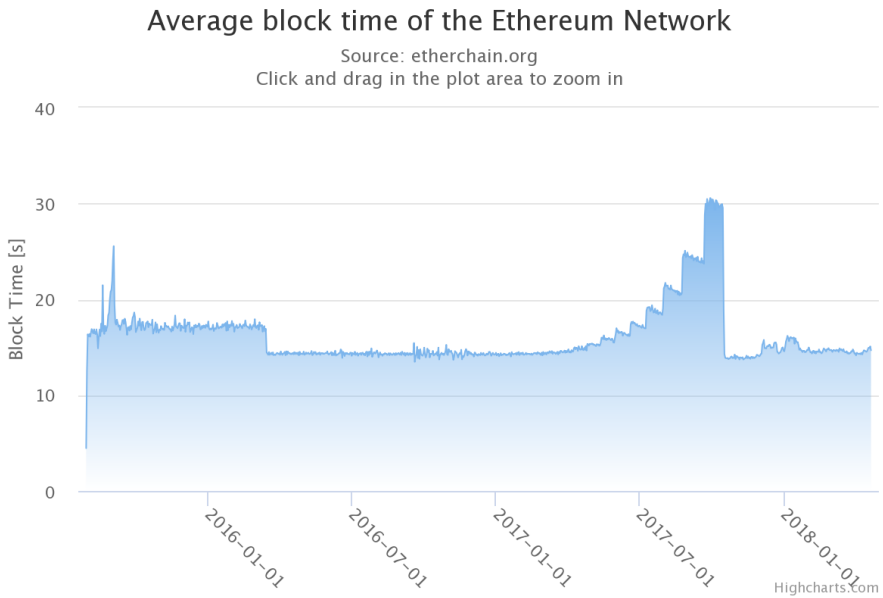
- **110mio ETH supply (stats)**
- **Increasing difficulty**
  - Mining in pools
  - Ethermine + Sparkpool ~55%





Blocktime and Gas

- **Block time: ~14-15s**
  - Ice age
- **Smart Contracts are turing complete**
  - Every instruction needs to be paid for (example)
  - Gas price ~13 gwei
- **Gas price / Gas limit by miners**
  - If you run out of gas, state is reverted, ETH gone



Appendix G. Fee Schedule

The fee schedule G is a tuple of 31 scalar values corresponding to the relative costs, in gas, of a number of abstract operations that a transaction may effect.

Name	Value	Description*
G_{zero}	0	Nothing paid for operations of the set W_{zero}.
G_{base}	2	Amount of gas to pay for operations of the set W_{base}.
$G_{verylow}$	3	Amount of gas to pay for operations of the set $W_{verylow}$.
G_{low}	5	Amount of gas to pay for operations of the set W_{low}.
G_{mid}	8	Amount of gas to pay for operations of the set W_{mid}.
G_{high}	10	Amount of gas to pay for operations of the set W_{high}.
$G_{extcode}$	700	Amount of gas to pay for operations of the set $W_{extcode}$.
$G_{balance}$	400	Amount of gas to pay for a BALANCE operation.
G_{sload}	200	Paid for a SLOAD operation.
$G_{jumpdest}$	1	Paid for a JUMPDEST operation.
G_{sset}	20000	Paid for an SSTORE operation when the storage value is set to non-zero from zero.
G_{sreset}	5000	Paid for an SSTORE operation when the storage value's zeroness remains unchanged or is set to zero.
R_{sclear}	15000	Refund given (added into refund counter) when the storage value is set to zero from non-zero.
$R_{suicide}$	24000	Refund given (added into refund counter) for suiciding an account.
$G_{suicide}$	5000	Amount of gas to pay for a SUICIDE operation.
G_{create}	32000	Paid for a CREATE operation.
$G_{codedeposit}$	200	Paid per byte for a CREATE operation to succeed in placing code into state.
G_{call}	700	Paid for a CALL operation.
$G_{callvalue}$	9000	Paid for a non-zero value transfer as part of the CALL operation.
$G_{callstipend}$	2300	A stipend for the called contract subtracted from $G_{callvalue}$ for a non-zero value transfer.
$G_{newaccount}$	25000	Paid for a CALL or SUICIDE operation which creates an account.
G_{exp}	10	Partial payment for an EXP operation.
$G_{expbyte}$	10	Partial payment when multiplied by $\lceil \log_{256}(exponent) \rceil$ for the EXP operation.
G_{memory}	3	Paid for every additional word when expanding memory.
$G_{txcreate}$	32000	Paid by all contract-creating transactions after the *Homestead transition*.
$G_{txdatazero}$	4	Paid for every zero byte of data or code for a transaction.
$G_{txdatanonzero}$	68	Paid for every non-zero byte of data or code for a transaction.
$G_{transaction}$	21000	Paid for every transaction.
G_{log}	375	Partial payment for a LOG operation.
$G_{logdata}$	8	Paid for each byte in a LOG operation's data.
$G_{logtopic}$	375	Paid for each topic of a LOG operation.
G_{sha3}	30	Paid for each SHA3 operation.
$G_{sha3word}$	6	Paid for each word (rounded up) for input data to a SHA3 operation.
G_{copy}	3	Partial payment for *COPY operations, multiplied by words copied, rounded up.
$G_{blockhash}$	20	Payment for BLOCKHASH operation.

$W_{zero} = \{\text{STOP, RETURN}\}$
$W_{base} = \{\text{ADDRESS, ORIGIN, CALLER, CALLVALUE, CALLDATASIZE, CODESIZE, GASPRICE, COINBASE, TIMESTAMP, NUMBER, DIFFICULTY, GASLIMIT, POP, PC, MSIZE, GAS}\}$
$W_{verylow} = \{\text{ADD, SUB, NOT, LT, GT, SLT, SGT, EQ, ISZERO, AND, OR, XOR, BYTE, CALLDATALOAD, MLOAD, MSTORE, MSTORE8, PUSH*, DUP*, SWAP*}\}$
$W_{low} = \{\text{MUL, DIV, SDIV, MOD, SMOD, SIGNEXTEND}\}$
$W_{mid} = \{\text{ADDMOD, MULMOD, JUMP}\}$
$W_{high} = \{\text{JUMPI}\}$
$W_{extcode} = \{\text{EXTCODESIZE}\}$

Blocktime and Gas

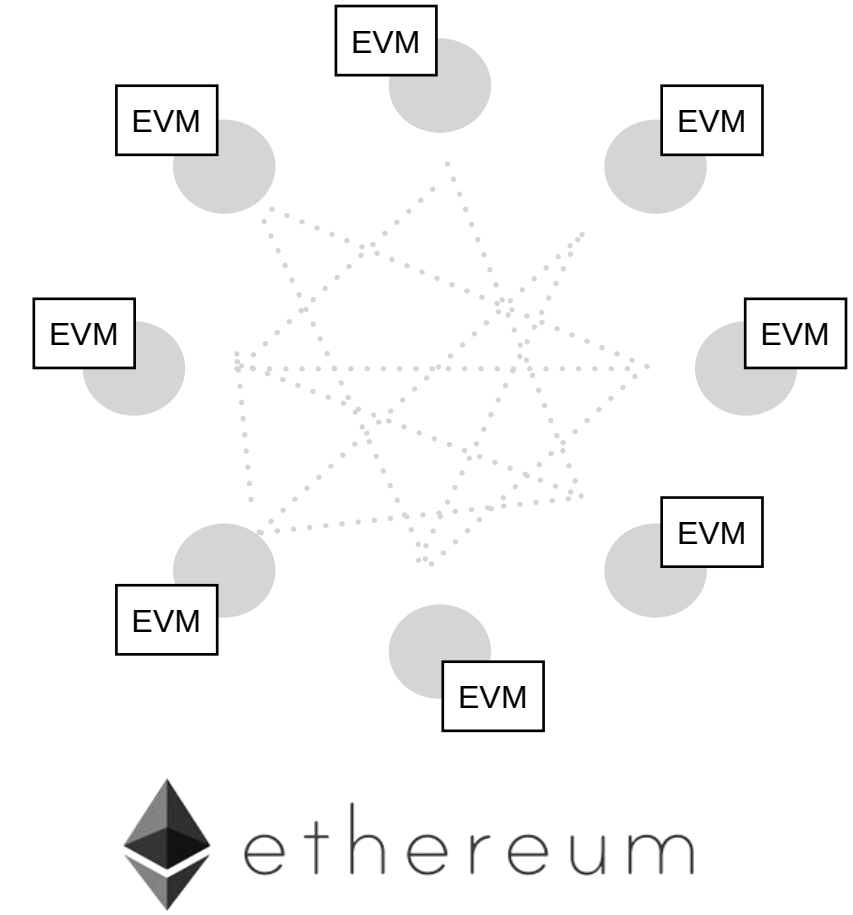
- **Gas Price set by Miner**
  - Gas price ~13 gwei
- **Miner decides which transaction at which gas price to include**
  - Market for TX
- **Gas price too low, longer waiting time until TX will be included**

- **Units:**

1 ether =	
1000000000000000000	wei
1000000000000000	Kwei
1000000000000	Mwei
1000000000	Gwei
1000000	szabo
1000	finney
1	ether
0.001	Kether
0.000001	Mether
0.000000001	Gether
0.000000000001	Tether

Ethereum smart contract

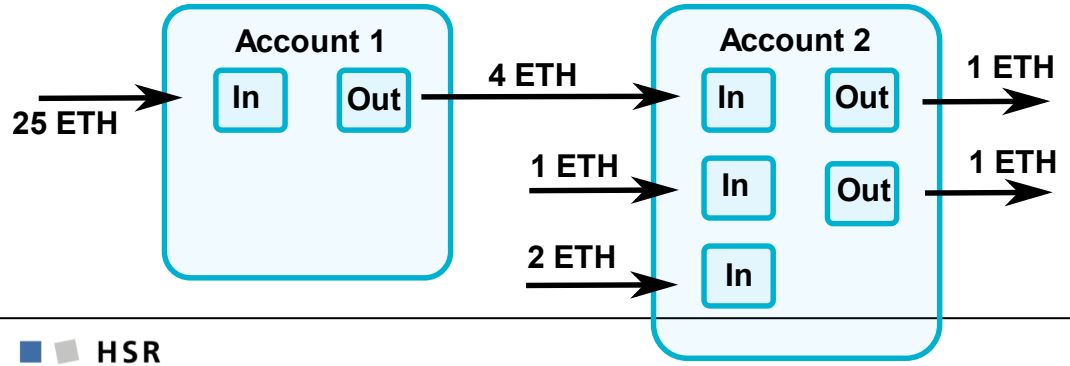
- **Computation on EVM is "very expensive“: every contract is run on every full Ethereum node**
  - Result on every node is the same
  - Global computer, always running, always correct
- **For now, proof of work (PoW)**
  - But difficult with ASIC (memory-hard), starts at 1GB RAM for full node and miner (2GB RAM should be enough)
- **Account-based**
  - 2 types: externally controlled, contract
    - Both can have and send ether
    - External accounts: controlled by private keys
  - Contract accounts never executed on their own
    - Contract accounts: controlled by code
    - All action fired from externally controlled accounts



Account vs UTXO - Introduction

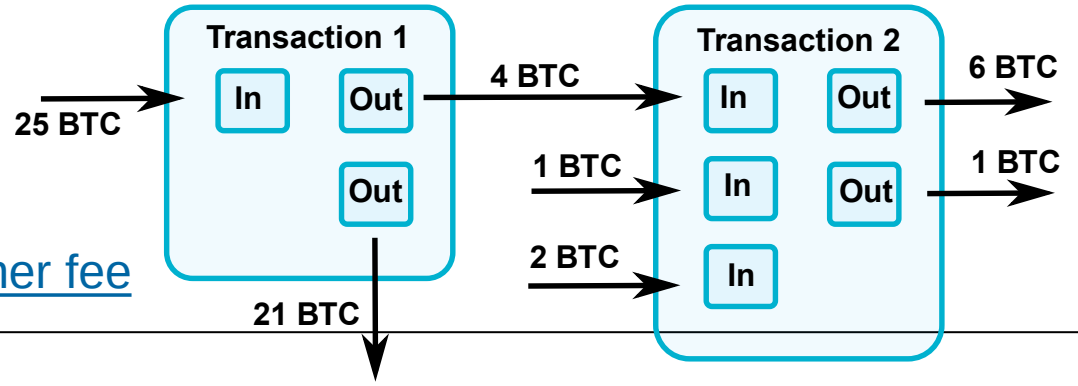
Account-based

- Global state stores a list of accounts with balances and code
- Transaction is valid if the sending account has enough balance
  - Balance on sender is deducted, new balance
- If the receiving account has code, the code runs, and state may be changed
  - Signature must match sending account



UTXO-based

- Every referenced input must be valid and not yet spent
- Total value of the inputs must equal or exceed the total value of the outputs
  - You always spend all outputs
- Transaction must have a signature matching the owner of the input for every input
  - Script determines if input is valid



Account vs UTXO – Pros and Cons

Account-based

- Large space savings
  - One input – one output – one signature
- Simplicity
  - Easier to code and understand
  - Easier for smart contracts (stateful scripting language)

UTXO-based

- Higher degree of privacy
  - New address for each TX
- No replay attacks – no nonce required
- Allows transactions to be processed in parallel
  - If outputs / inputs are separate

Solidity - http://solidity.readthedocs.io

- **Version Pragma - reject compiled with specific compiler versions**

```
pragma solidity ^0.4.23; //not before 0.4.23, not after 0.5.0
```

- **Comments**

```
// This is a single-line comment.
/*
This is a
multi-line comment.
*/
```

- **Contract with State Variables (state change is expensive!)**

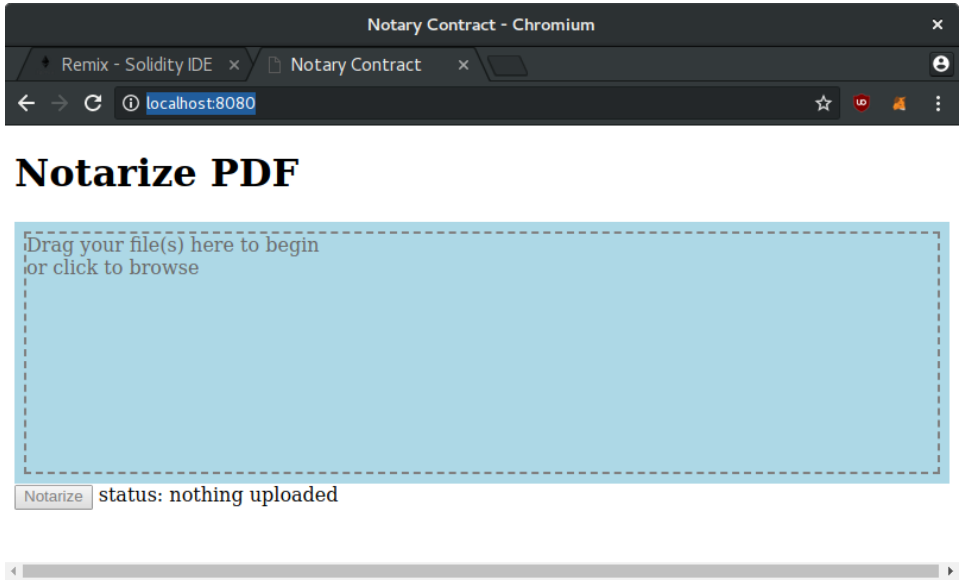
```
pragma solidity ^0.5.7;
//minimal contract
contract Example1 {
    uint256 counter;
}
```

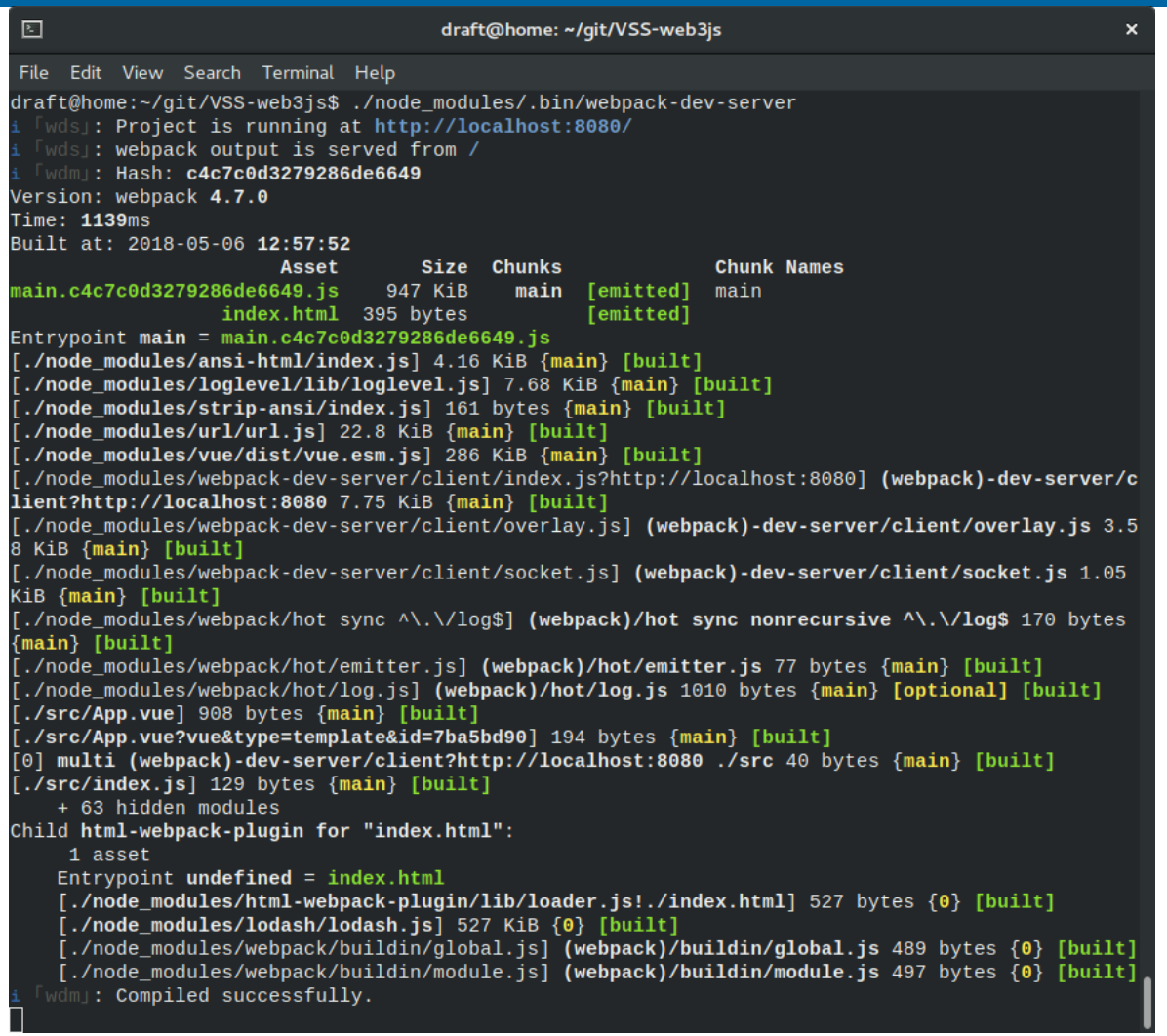
https://learnxinyminutes.com/docs/solidity/

https://solidity.readthedocs.io/en/v0.5.7/

Example

- **Installation**
  - yarn install
  - ./node_modules/.bin/webpack-dev-server
- **Open Browser: http://localhost:8080/**





Questions?

Dr. Thomas Bocek
thomas.bocek@hsr.ch