FS19

BLOCKCHAIN, BITCOIN, ETHEREUM III

Ethereum Smart Contracts

T. Bocek

thomas.bocek@hsr.ch

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Agenda

- **Blockchain, Bitcoin, Ethereum, continued**
- **Ethereum**
  - Introduction
  - Statistics
- **Solidity**
  - Examples

Distributed Systems - In the *News*

- **18.05.2019** **Number go down — the single trade that crashed Bitcoin [heise]**
  - Reason: capital flight from China?
  - BTC around 8000, 17.05.2019, to 6000
  - Single sale of 5000BTC on Bitstamp
  - Margin trading: using borrowed funds, used as collateral - magnify gains and loss
  - BitMEX is offering margin trading on BTC/USD, but does not trade BTC/USD directly
    - Uses price from Bitstamp + Coinbase
    - Coincidence?

- **14.05.2019** **Employing QUIC Protocol to Optimize Uber's App Performance**
  - Apps relying entirely on wireless connectivity
  - HTTP/2 works poorly in dynamic, lossy wireless networks due to TCP
  - TCP was designed for wired networks
  - TCP: lost packet → congestion
    - But could be bad wireless connectivity
    - → Do not reduce congestion window, but resend immediately
  - QUIC/HTTP3 reduced latency by 50%
  - Fallback to TCP: unknown if UDP failures or poor network conditions

Distributed Systems - In the *News*

- **15.05.2019 China blockiert Wikipedia in sämtlichen Sprachen**
  - “Kurz vorm Jahrestag des Tiananmen-Massakers sperren Chinas Zensoren die Online-Enzyklopädie Wikipedia.”
  - China blocked Wikipedia in Chinese language, but now in all languages
  - Also blocked: Facebook, Twitter, YouTube, Google services, WhatsApp, New York Times, or Wall Street
    - My visit to Shanghai: although I had Internet I felt lost, needed to use Baidu, Didi, and WeChat

- **Email: https://nusenu.github.io/OrNetStats/**
  - Tor Relay Operators in End-to-End Correlation Position
  - Alice protection is lost should her Tor client use relays controlled by a single entity
    - Distinct subnets – attacker can have multiple subnets
    - “Family” – declaring your group of relays as belonging together – attacker can have different families
    - Entry / exit flags
  - Top 10 Autonomous System Names by CW Fraction
    - Hetzner, OVH
  - OS Distribution (Relays)
    - Linux, BSD

Web Engineering - In the *News*

- https://hackathon.trustsquare.ch/

Web Engineering - In the *News*

Learning Goals

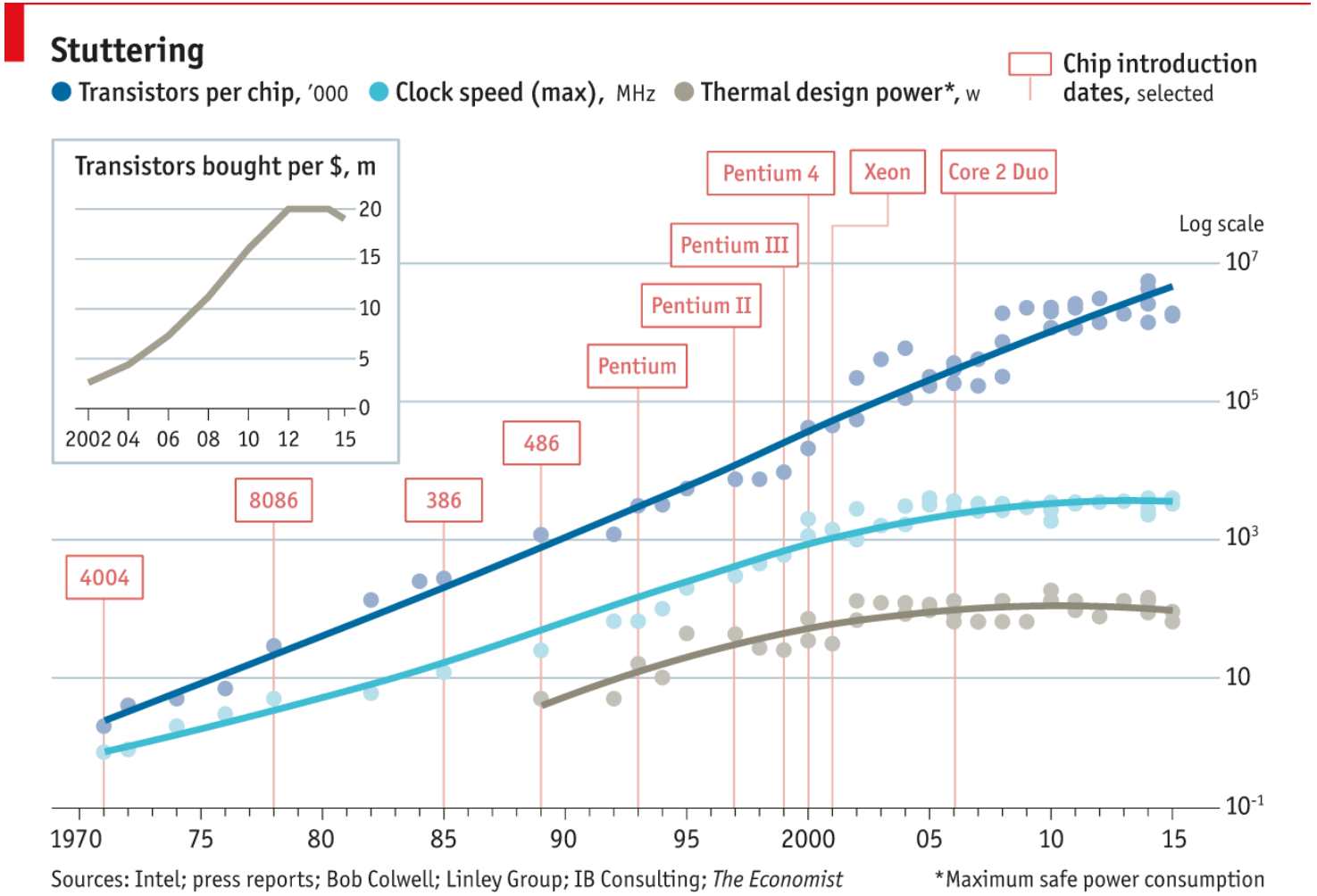
- **I can describe and explain a 51% attack**
- **I know the difference between Ethereum and Bitcoin**
- **I can write and deploy a simple smart contract**

URL: b.socrative.com/student/

Room: HSR

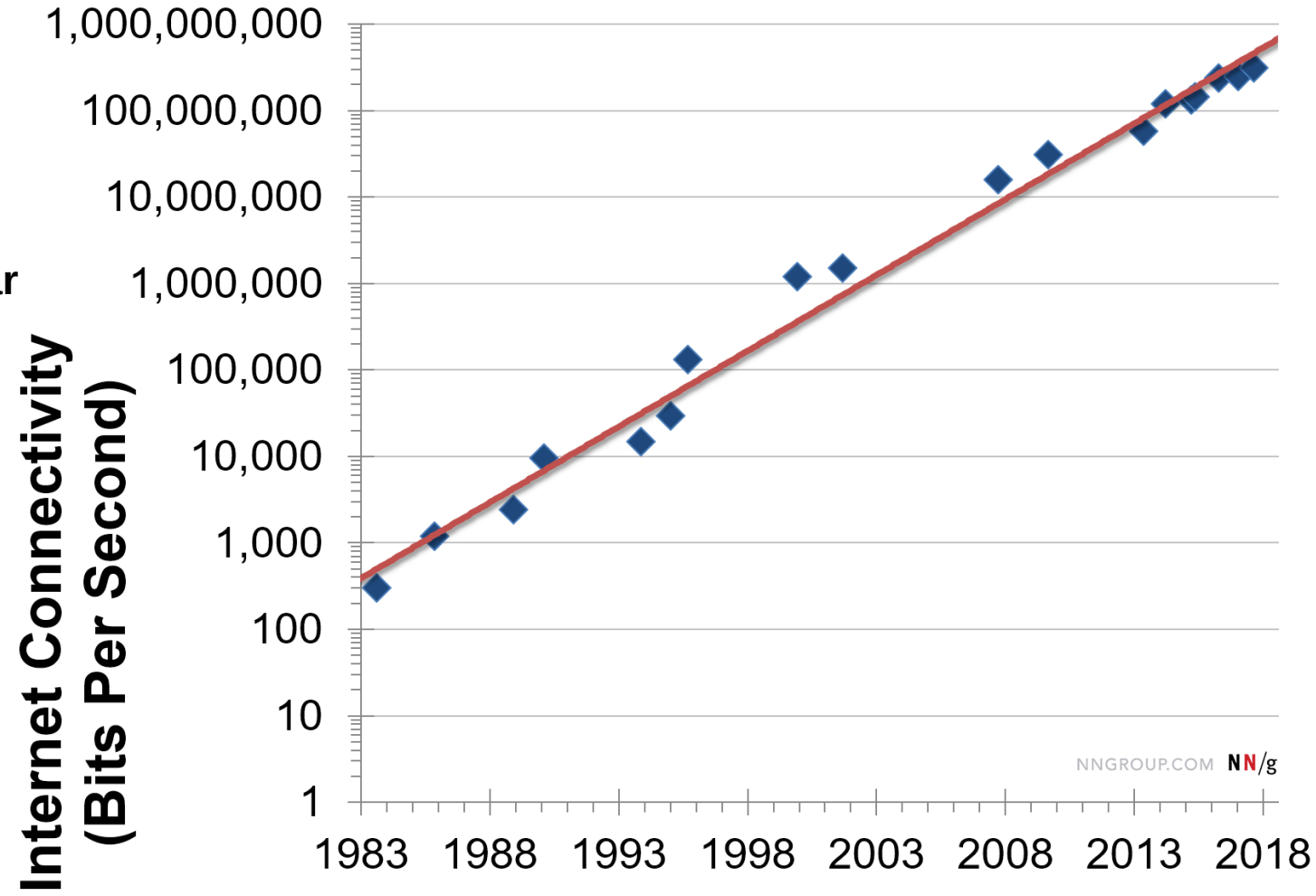
Trends (1)

- **Moore's Law – nr. of transistors doubles every 18 month**
- **Still alive…**
- **But dead in 2021?**



Trends (2)

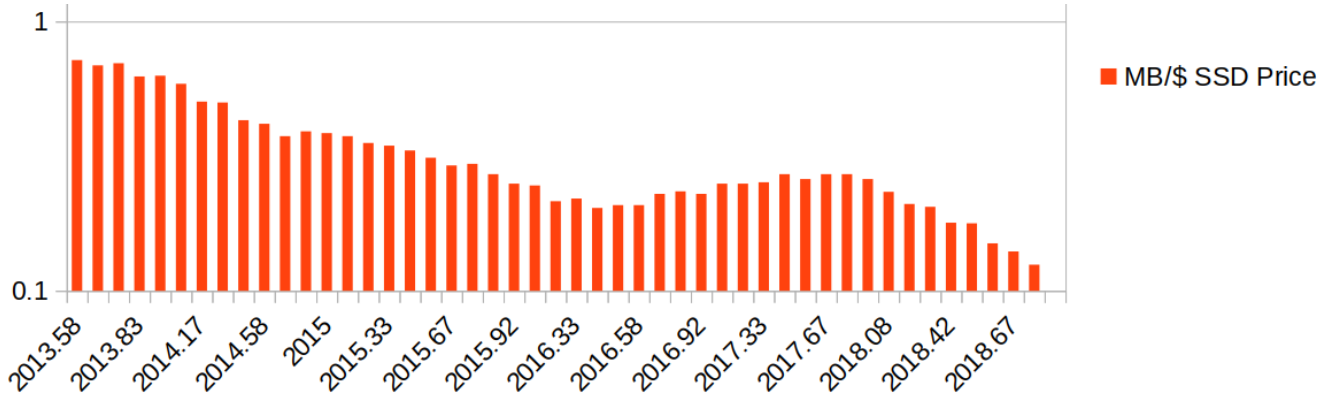
- **Nielsen's Law: a high-end user's connection speed grows by 50% per year**
- **Bandwidth grows slower than computer power**
  1. Telecoms companies are conservative
  2. Users are reluctant to spend much money on bandwidth
  3. The user base is getting broader
- **Optimize for bandwidth**
- **Zmap complete scan of the IPv4 address space in under 5 minutes**



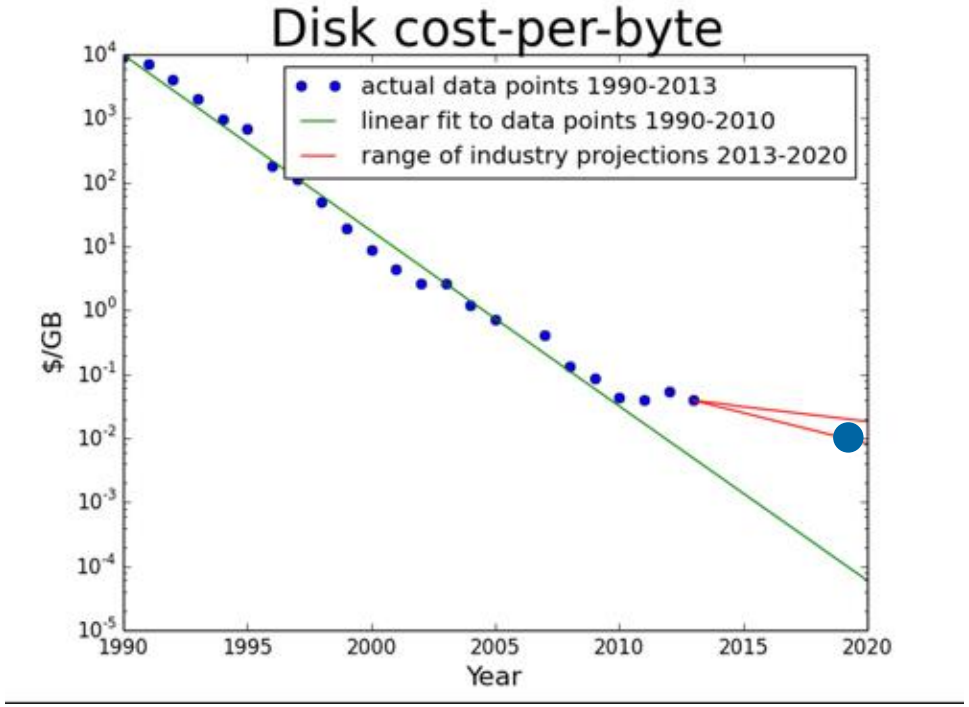
		Annualized Growth Rate	Compound Growth Over 10 Years
Nielsen's law	Internet bandwidth	50%	57×
Moore's law	Computer power	60%	100×

Trends (3)

- **Kryder's Law: disk density doubling every 13 month**
- **«Soon hard drives will migrate into phones, still cameras, PDAs, cars and everyday appliances»**
  https://www.scientificamerican.com/article/kryders-law/ , **Aug. 2005**
- **User behavior changed**
  - SSD, speed is important
- **Cloud – Dropbox, Spotify**
  - Streaming



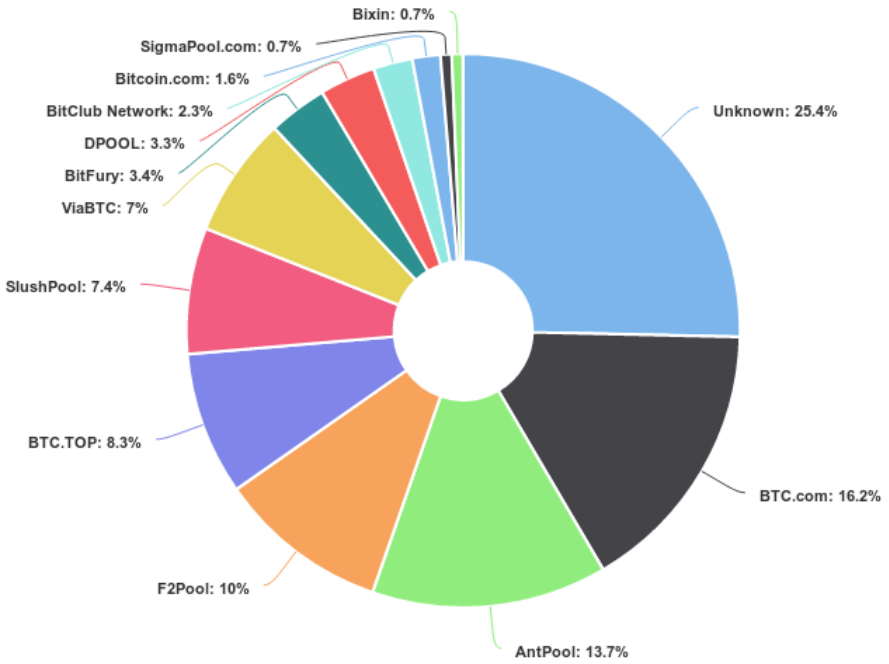
Source: https://jcmit.net/flashprice.htm



http://blog.dshr.org/2016/05/the-future-of-storage.html

51% Attack

- **Control over 50% of the scarce resources**

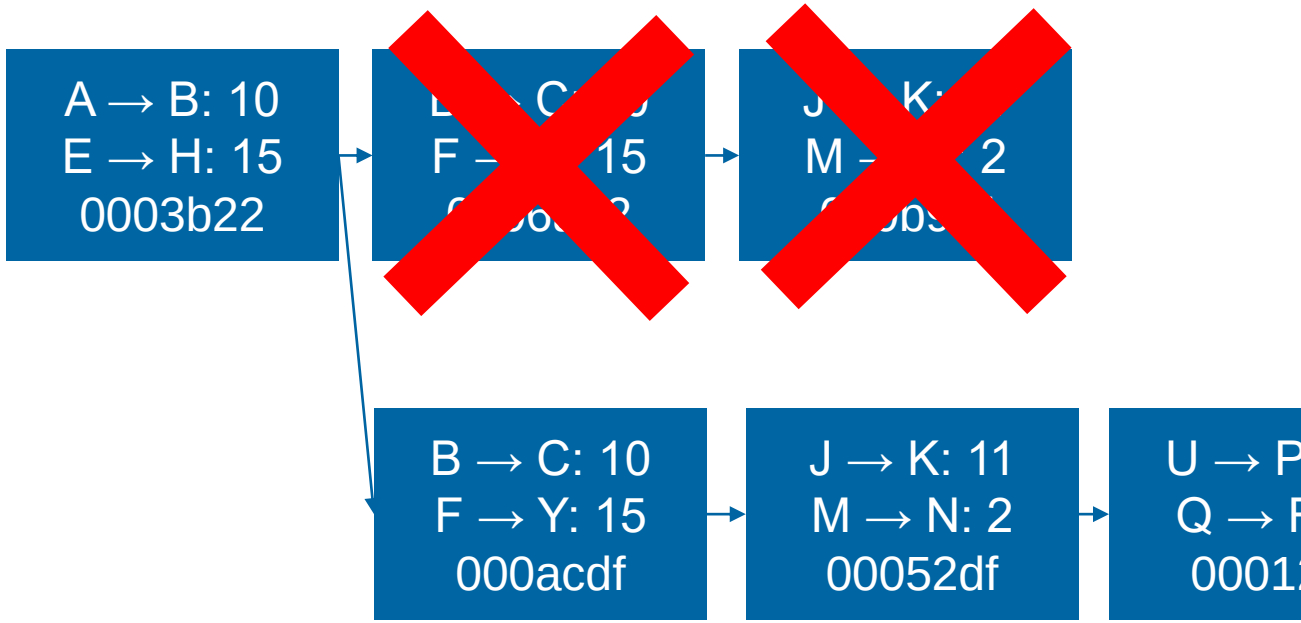


http://blockchain.info/pools

- **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)."

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

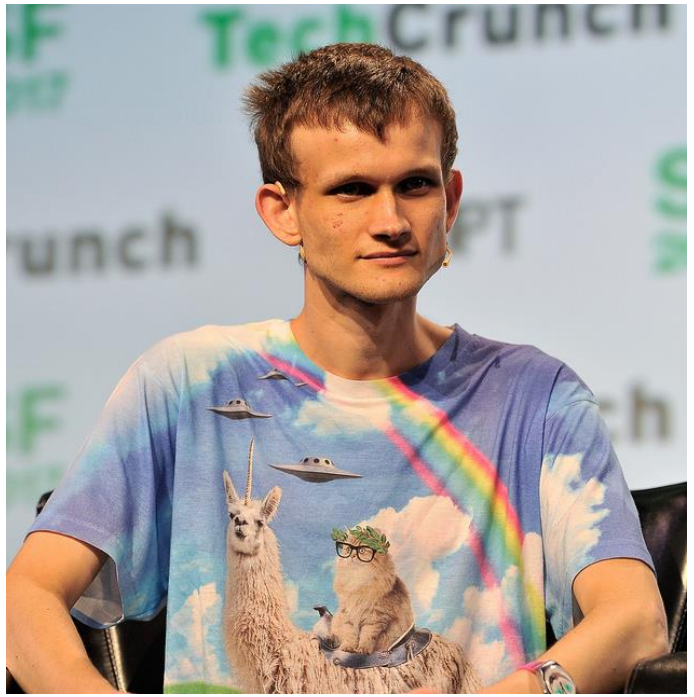


URL: b.socrative.com/student/

Room: HSR

Bitcoin / Ethereum

- **“Issues” with Bitcoin**
  - Slow development, implementing new features difficult
    - SegWit vs. BTU (segregated witness vs. increasing block size voting) Cash vs. SV
  - Bitcoin Script limited
    - Lightning network
- **Ethereum (1 ETH ~252$)**
  - 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

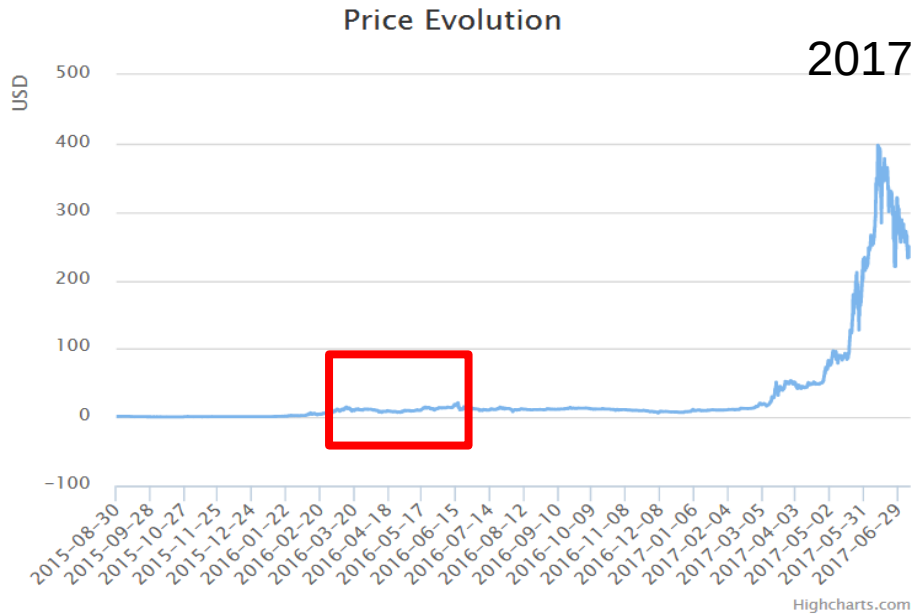
- **Olympic (past) – released 09.05.2015**
  - Last Ethereum Proof-of-Concept series
  - “Olympic will feature a total prize fund of up to 25,000 ether” (now 3.5m USD)
- **Frontier (past) - released 30.07.2015**
  - Main public network, “Beta”/use at your own risk
- **Homestead (past) - released 14.03.2016**
  - Public network considered “stable”, integrate critical protocol changes
- **Metropolis (Byzantium, past) - released 16.10.2017**
  - Initial supports for ZKP (private smart contracts), reducing mining reward to 3 ETH
- **Metropolis (Constantinople, now) – released 27.2.2019**
  - Bitshifting, mining reward 2 ETH (delay difficulty bomb), other optimizations
- **Serenity (future)**
  - Scalability, proof-of-stake, Sharding, ZKP

URL: b.socrative.com/student/

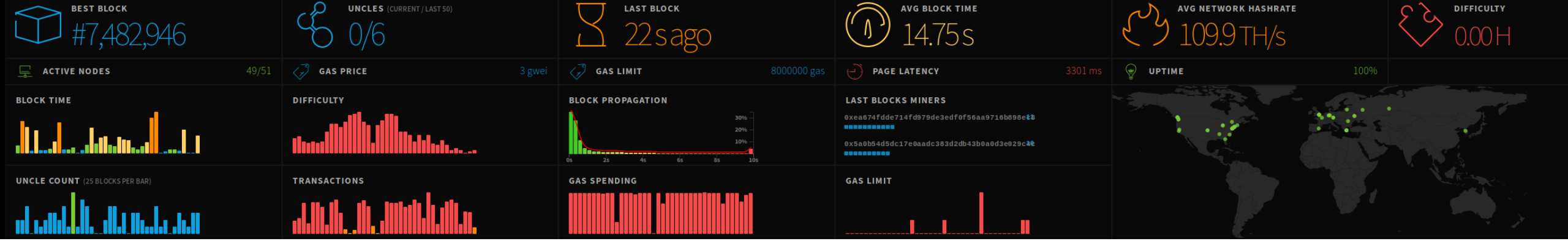
Room: HSR

Ethereum Stats

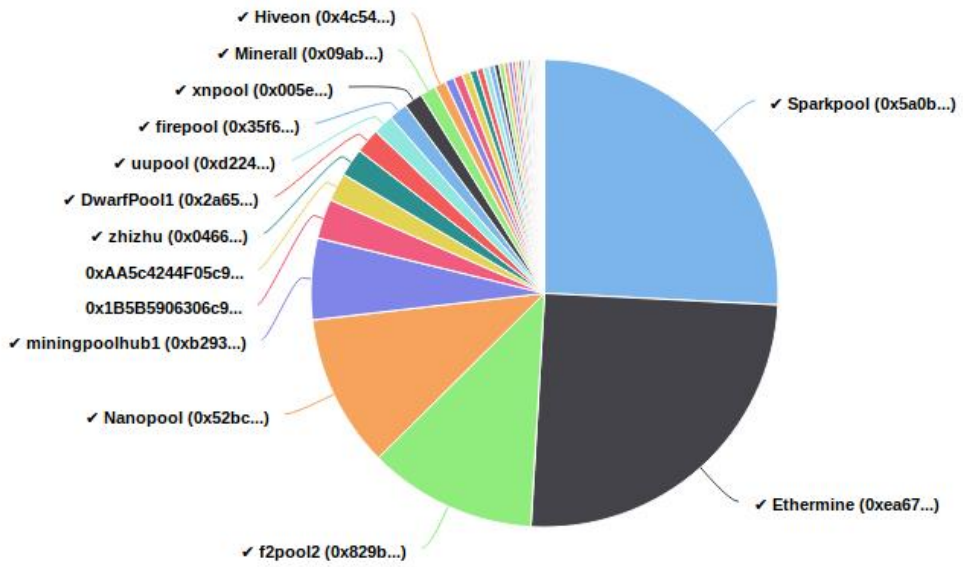
- **Basic Stats**
  - 2nd in market cap ~ 26b USD
  - Transactions (highest ~15.5 TPS) now ~630’000 per day
  - Node count (~11’000 nodes)
  - Blocksize ~20KB
  - Accounts (58mio)

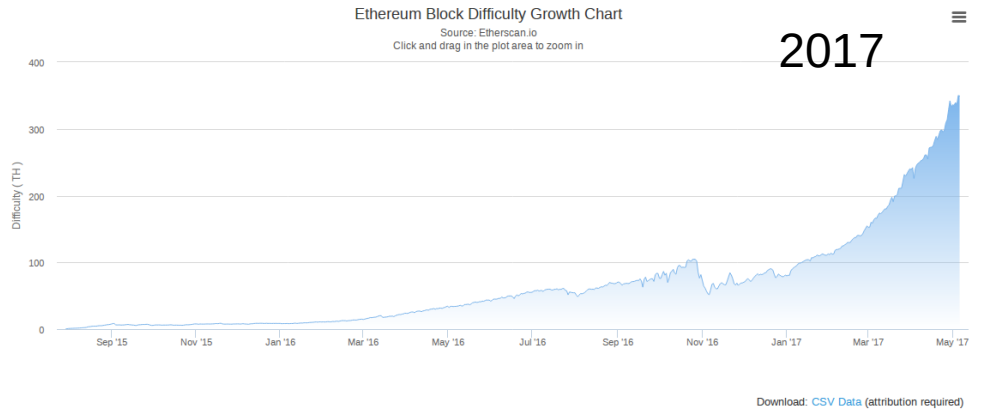


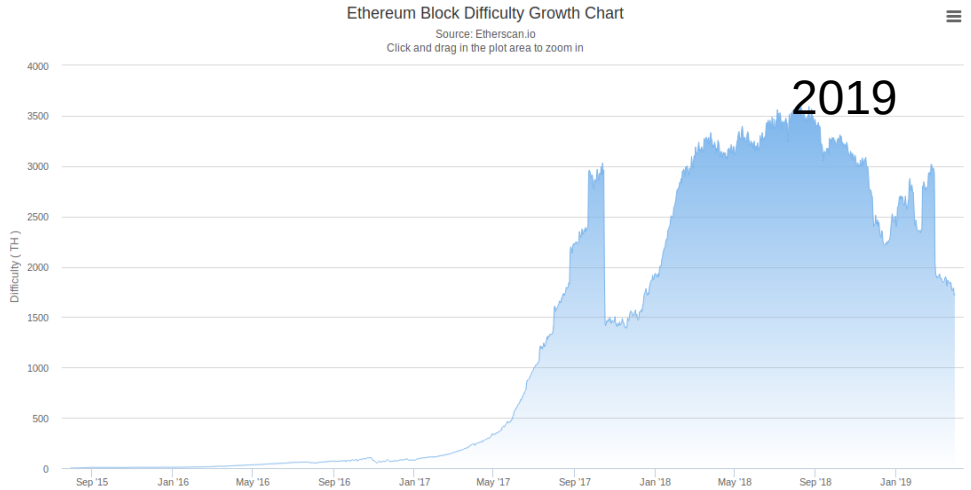
HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL
FHO Fachhochschule Ostschweiz



- **106mio ETH supply (stats)**
- **Increasing difficulty**
  - Mining in pools
  - Ethermine + Sparkpool 50.9%

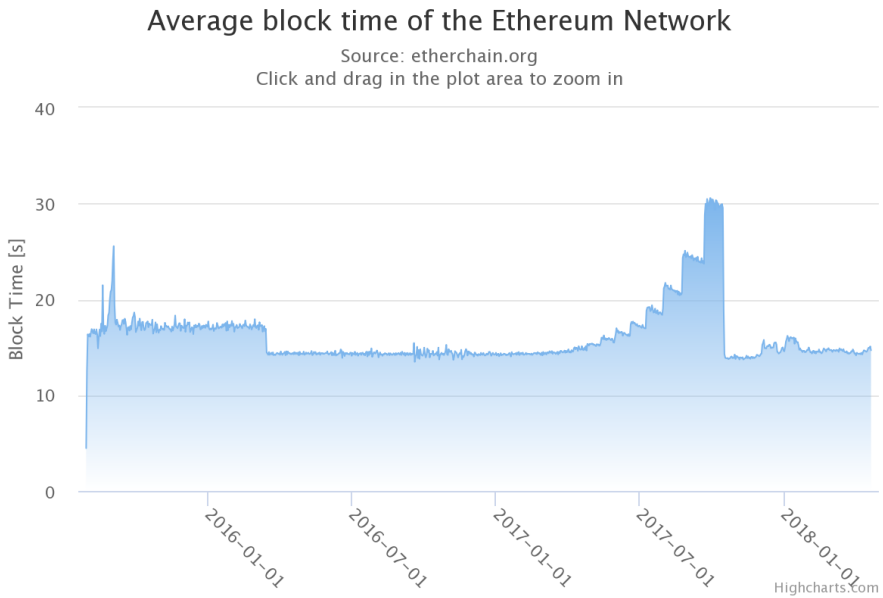






Blocktime and Gas

- **Block time: ~14-15s**
  - Ice age
- **Contracts are turing complete**
  - Every instruction needs to be paid for (example)
  - Gas price ~2 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.

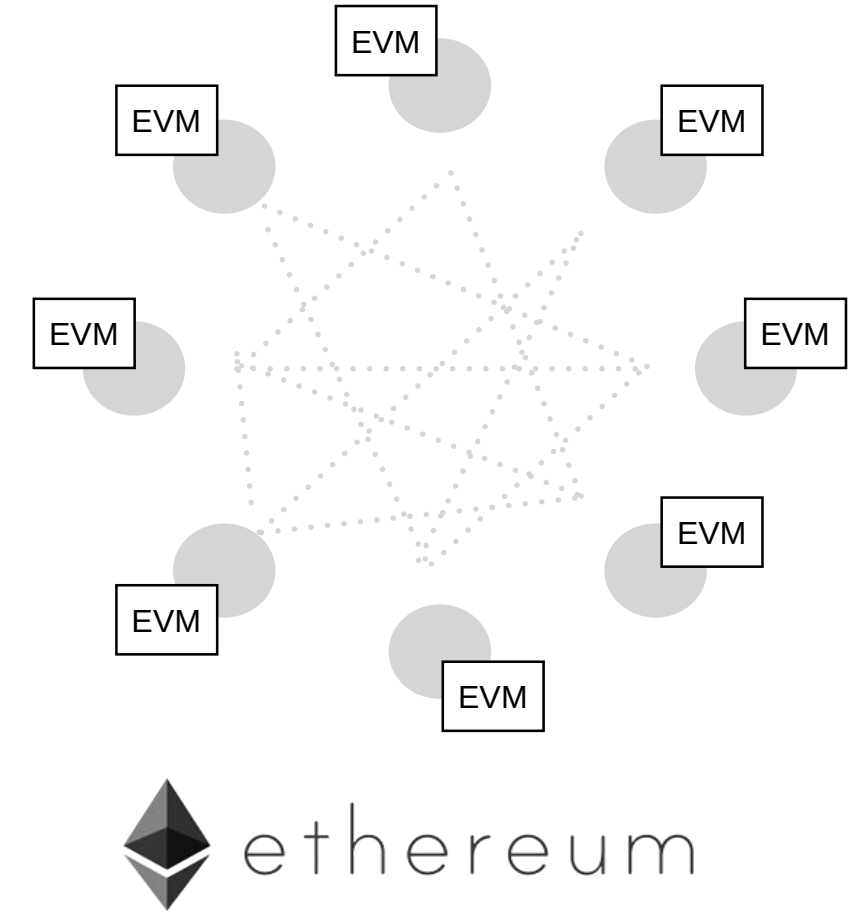
$W_{zero} = \{\text{STOP, RETURN}\}$
$W_{base} = \{\text{ADDRESS, ORIGIN, CALLER, CALLVALUE, CALLDATASIZE, CODESIZE, GASPRICE, COINBASE, TIMESTAMP, NUMBER, DIFFICULTY, GASLIMIT, POP, PC, MSIZE, GAS}\}$
$W_{verylow} = \{\text{ADD, SUB, NOT, LT, GT, SLT, SGT, EQ, ISZERO, AND, OR, XOR, BYTE, CALLDATALOAD,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

URL: b.socrative.com/student/

Room: HSR

Ethereum smart contract

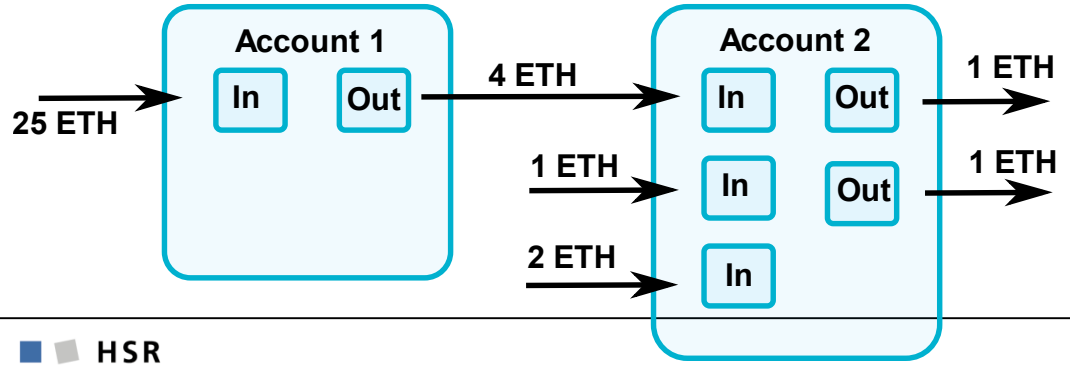
- **Computation on EVM is "very expensive“: every contract is run on every full Ethereum node**
  - Result on every node is the same
  - Global computer, always running, always correct
- **For now, proof of work (PoW)**
  - But difficult with ASIC (memory-hard), starts at 1GB RAM for full node and miner (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

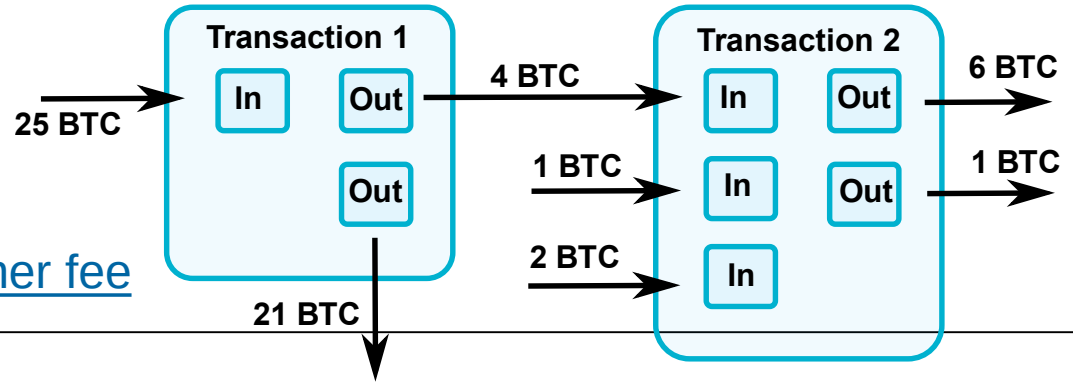
Account-based

- 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

URL: b.socrative.com/student/

Room: HSR

Solidity

Solidity -

- **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 (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/

Solidity - http://solidity.readthedocs.io

- **Types**
  - bool: true and false, int / **uint** (int8, int16, …, int256), **address**: 20 bytes, fixed size arrays: bytes1, bytes2, bytes3, ..., bytes32, variable sized: bytes (push), strings
- **Structs**

```
struct Account {
  string name;
  uint256 amount;
}
```

- **Mapping**

```
mapping (address => uint256) accounts; //mapping with basic types
mapping (uint256 => Account) accounts; //mapping with structs
```

Solidity – Basics

- **Arrays**

```
string[] names
uint256 newLength = names.push("John");
```

- **Another contract**

```
pragma solidity ^0.5.7;
contract Example2 {
    struct Account {
            string addr;
            uint256 amount;
    }
    uint256 public counter;
    mapping (uint256 => Account) accounts;
}
```

- **Create getter/setter**

```
function get(uint256 nr) public view returns (string) {
    return accounts[nr].addr;
}
function set(uint256 nr, string addr) public {
    require(owner == msg.sender);
    accounts[counter++] = Account(addr, nr);
}
```

- **Read state variables**
  - Constant function does not modify state
  - Reads “for free”

Solidity – ownership

- **Preconditions**
  - Special Variables and Functions

```
require(owner == msg.sender);
```

- **Set owner – old syntax**

```
pragma solidity ^0.4.0;
contract Example3 {
    address owner;
    function Example3() {
        owner = msg.sender;
    }
}
```

- **Constructor – new syntax**

```
constructor(string addr) public {
    owner = msg.sender;
}
```

- **Events – notifications**

```
pragma solidity ^0.5.7;
contract Example4 {
    event Message(string msg);
```

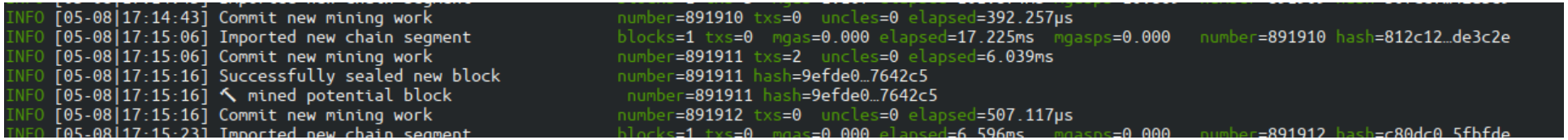
- **E.g. in the Geth console**

```
at =
browser_example1_sol_example2Contract.at
("0x…")

var event = at.Message(function(error,
result){console.log(result.args.msg)});
```

Solidity Deployment (with a full client)

- **Create address (geth) – or parity, or MEW, or…**
  - Ethereum address = create random private key → derive public key → hash it, take last 20bytes
    - geth --rinkeby account new
- **Important sites**
  - https://rinkeby.etherscan.io/ (blockchain explorer)



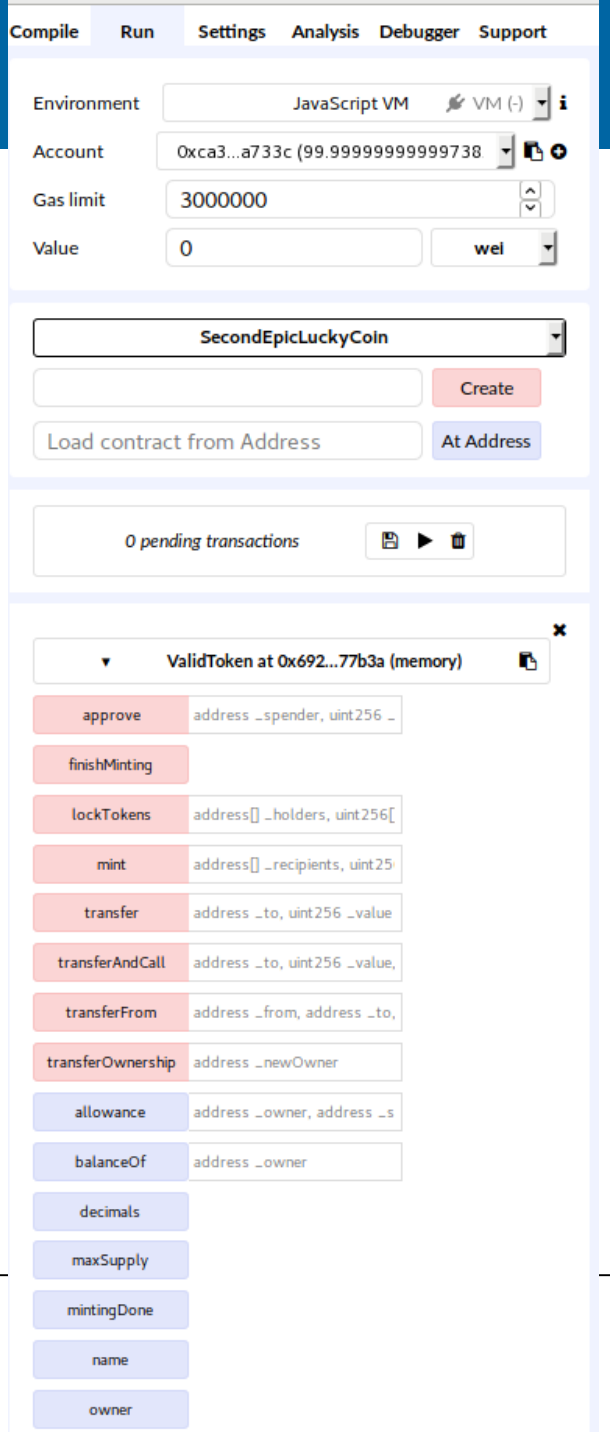
  - No mining (use twitter with https://www.rinkeby.io)
- **Start geth**
  - geth --rinkeby --syncmode light --rpc --rpccorsdomain '*' --rpcapi "eth,net,web3,personal" --unlock 0 --password <(echo 123456)
  - geth attach http://127.0.0.1:8545 or
  - Connect Remix Solidity IDE to geth / or use JavaScript Ethereum blockchain in Remix

Solidity IDE

- **Create your first contract**

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

- **Remix – Solidity IDE – Environment: Web3 Provider**
  - http://localhost:8545
- **«Deploy» contract**
- **Mixed content - workaround: use http**
  - **https://remix.ethereum.org**
- **Select Web3 Provider (geth is your Web3 Provider)**
  - Alternatively: use scripts, MetaMask, or JavaScript EVM



URL: b.socrative.com/student/

Room: HSR

Questions?

Prof. Dr. Thomas Bocek
thomas.bocek@hsr.ch