

FS19

BLOCKCHAIN, BITCOIN, ETHEREUM II

Ethereum Smart Contracts

T. Bocek

thomas.bocek@hsr.ch

Agenda

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

Distributed Systems - In the News

- **29.03.2019** [Bitcoin Price Hits 5-Week High With Chart Echoing 2015 Pre-Rally Pattern](#)
 - Here we go again 😊
- **01.04.2019** [Police Freeze Bank Accounts, Seize Luxury Cars in Probe of \\$22 Million ICO Promoter Vanbex](#)
 - 2017 initial coin offering (ICO) raised \$22 million
 - Token sale: [FUEL](#)
 - “Vanbex told investors that the token would be usable in a forthcoming smart contract system called Etherparty”
 - “No intention to build product, but use funds [for own personal benefit](#).” – no criminal charges filed
- **Recently discovered (13.11.2018):** [Let's Take a Crack at Understanding Distributed Consensus](#)
 - From “What is a Distributed System to “Proof-of-Work”
 - “.., Nakamoto Consensus is Byzantine fault-tolerant. But it clearly doesn't achieve consensus the way consensus researchers would traditionally assume.”
- **Announcement:** [NEO Blockchain Meetup](#), 4.4.2019

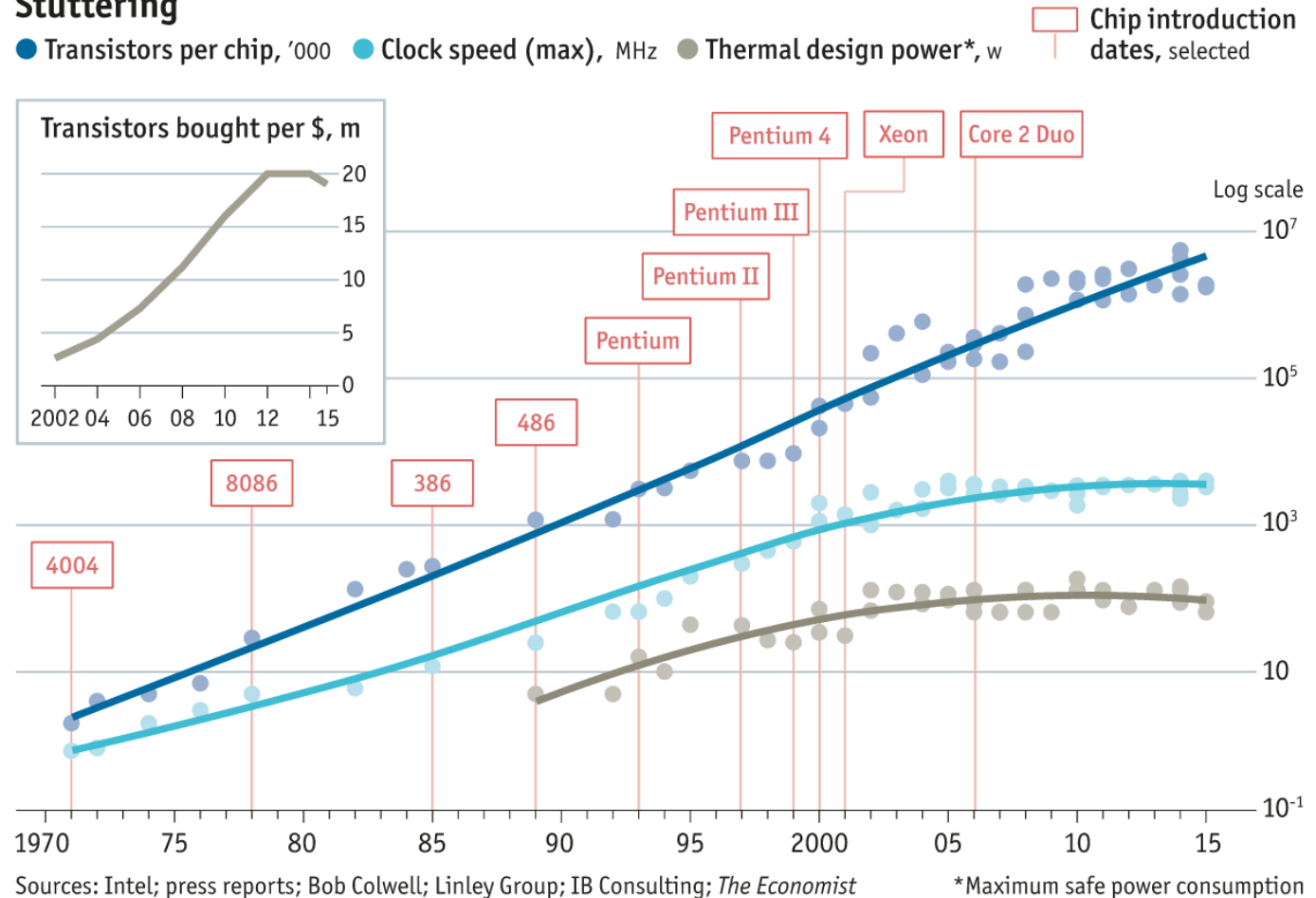
Learning Goals

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

Trends (1)

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

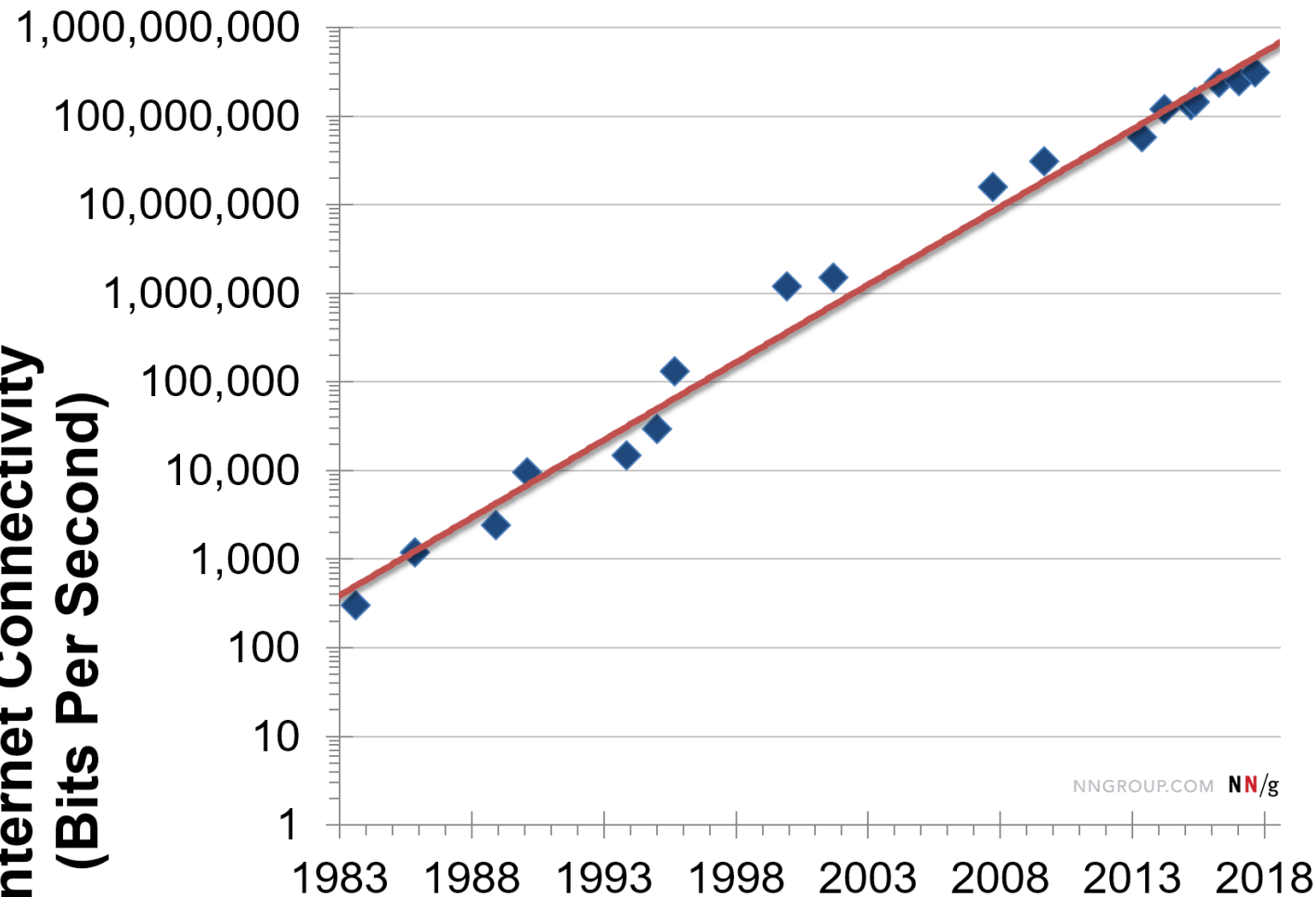
Stuttering



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**

Internet Connectivity
(Bits Per Second)



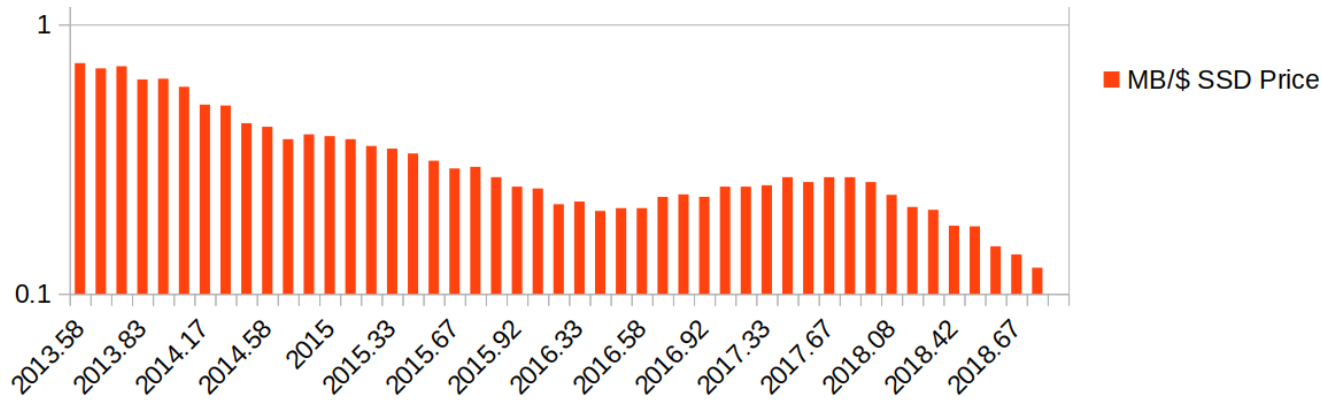
		Annualized Growth Rate	Compound Growth Over 10 Years
Nielsen's law	Internet bandwidth	50%	57×
Moore's law	Computer power	60%	100×

Trends (3)

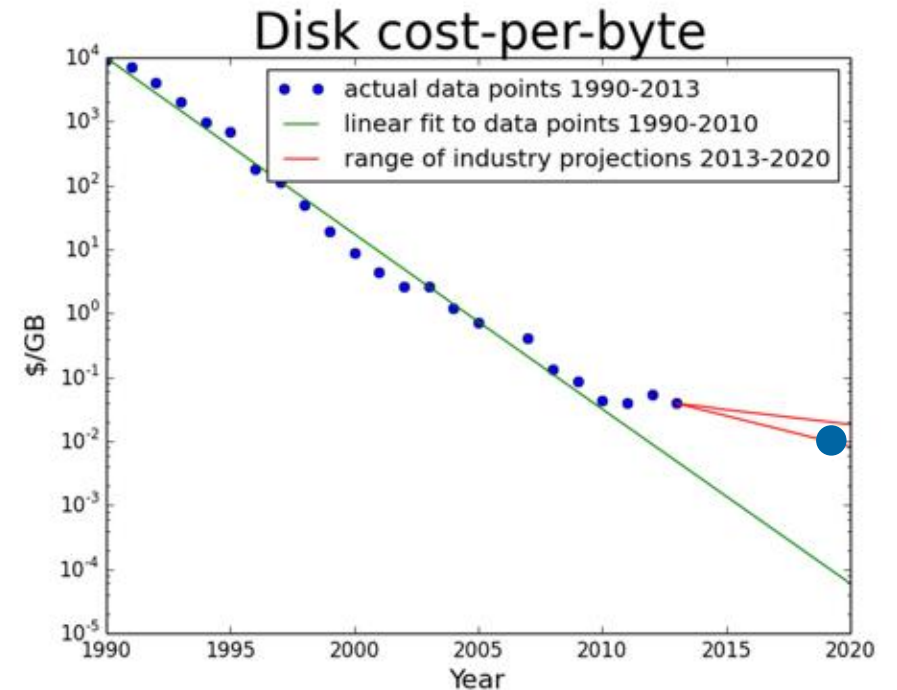
- Kryder's Law: disk density doubling every 13 month
- «Soon hard drives will migrate into phones, still cameras, PDAs, cars and everyday appliances»

<https://www.scientificamerican.com/article/kryders-law/> , Aug. 2005

- User behavior changed
 - SSD, speed is important
- Cloud – Dropbox, Spotify
 - Streaming



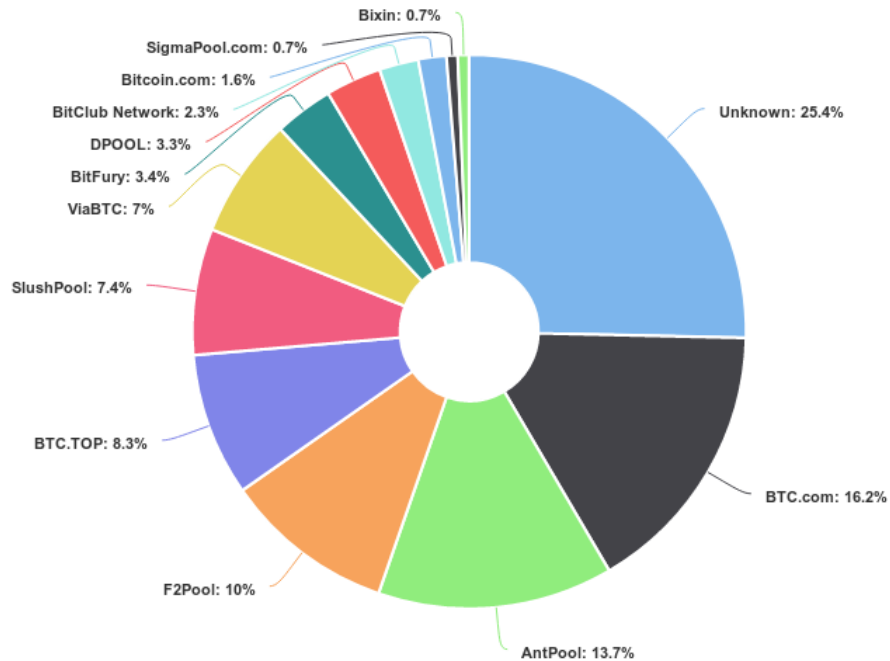
Source: <https://jcm.it.net/flashprice.htm>



<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>

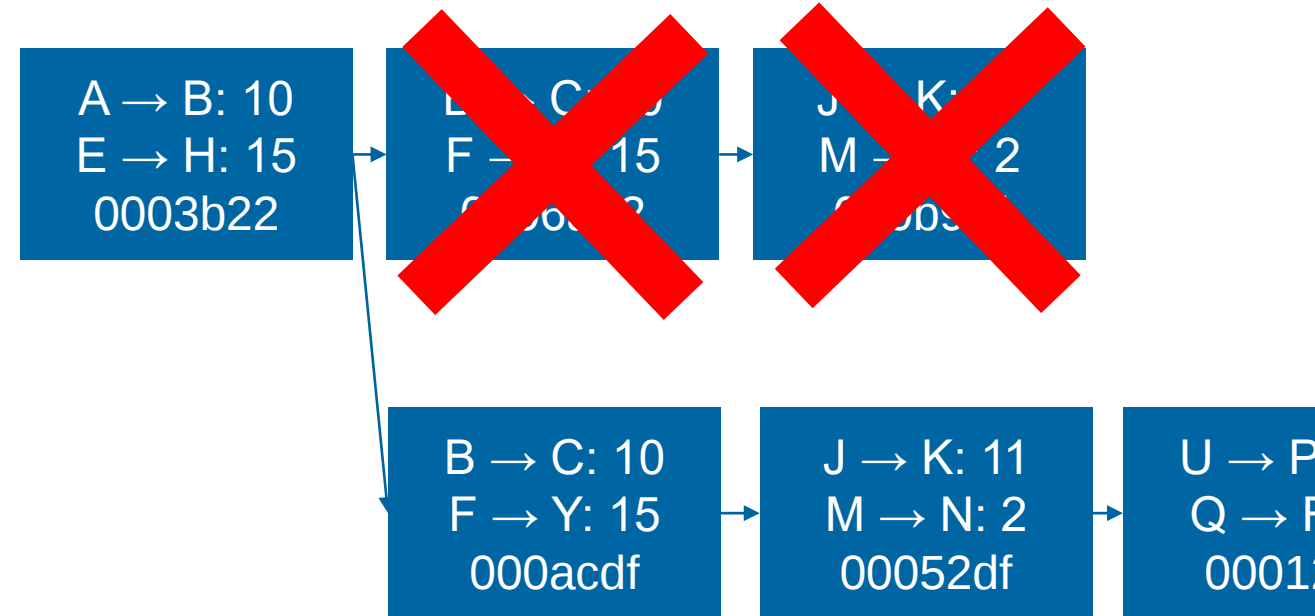
- PoW: majority of hashing power, PoS: majority of coins

- 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/

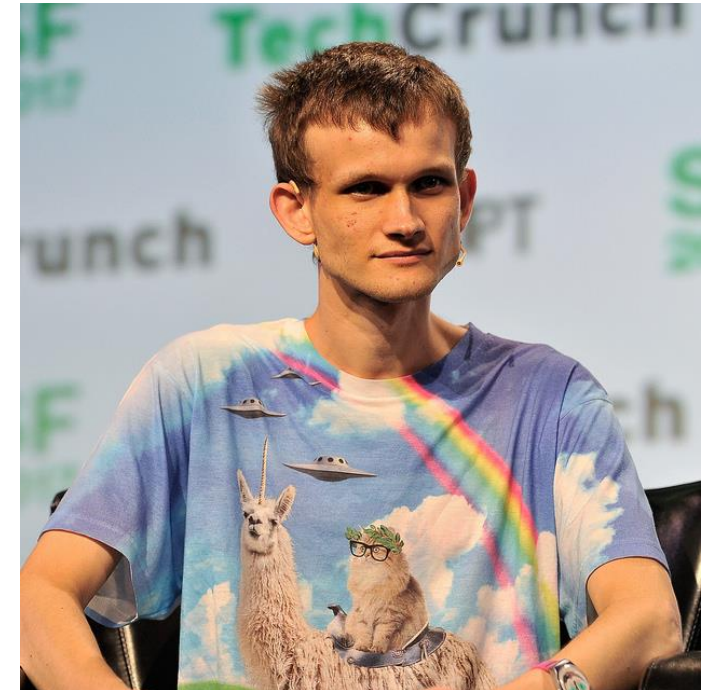
Room: HSR

■ “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 - [beta](#)

■ Ethereum ([1 ETH ~152\\$](#))

- Generalized blockchain (loops, arithmetics, 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

- **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 03.14.2016**
 - 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



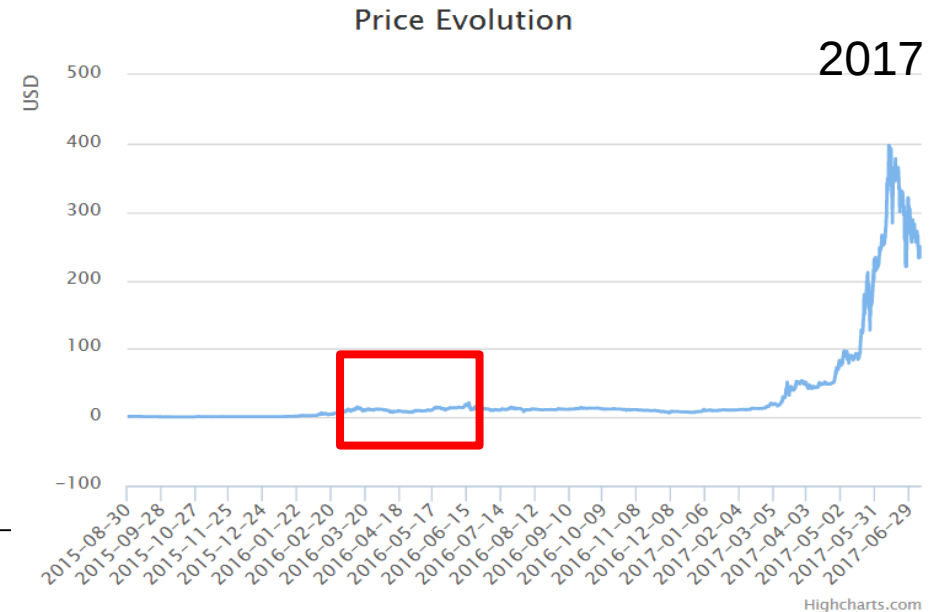
URL: b.socrative.com/student/

Room: HSR

Ethereum Stats

■ Basic Stats

- 2nd in [market cap](#) ~ 16b USD
- Transactions (highest ~15.5 TPS)
now ~[630'000 per day](#)
- [Node count](#) (~11'000 nodes)
- [Blocksize](#) ~20KB
- [Accounts](#) (58mio)

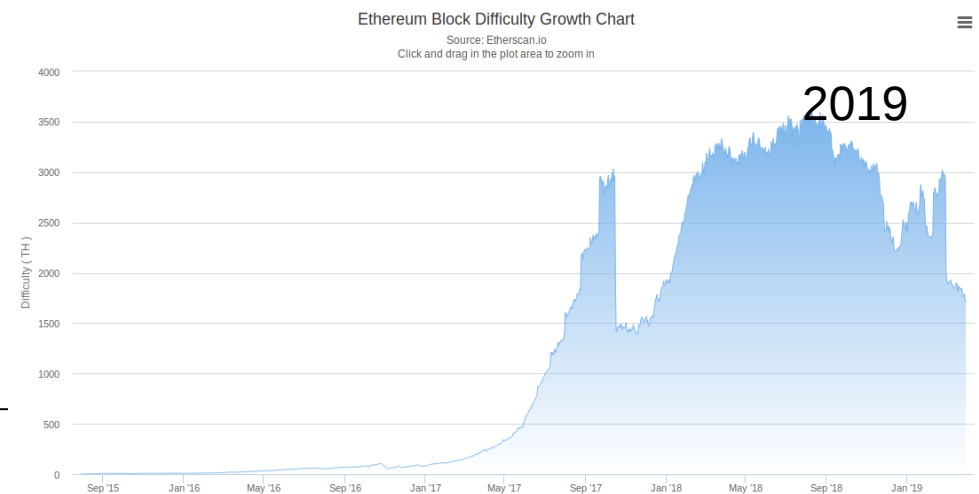
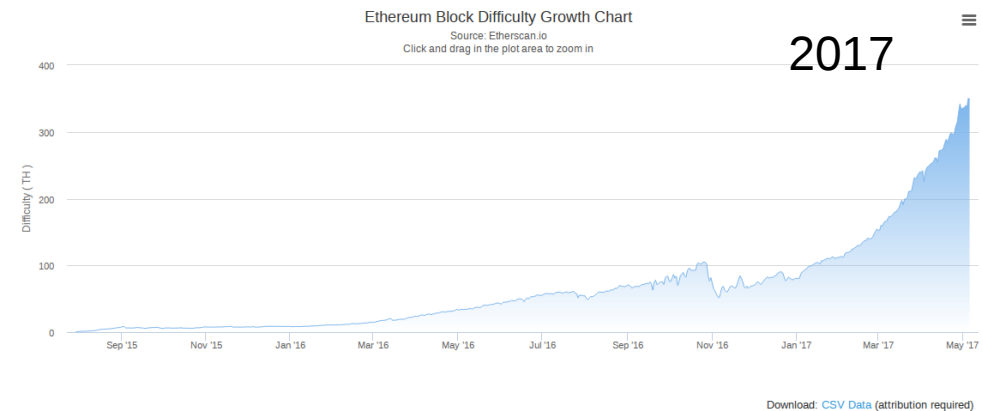
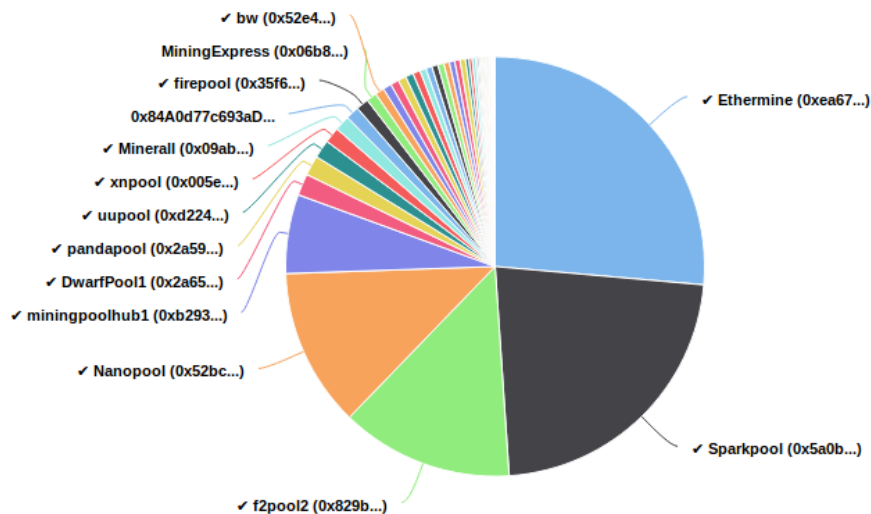




■ 105mio ETH supply ([stats](#))

■ Increasing difficulty

- [Mining in pools](#)
- Ethermine + Sparkpool 48.8%



Blocktime and Gas

■ Block time: ~14-15s

■ [Ice age](#)

■ Contracts are turing complete

■ Every instruction needs to be paid for ([example](#))

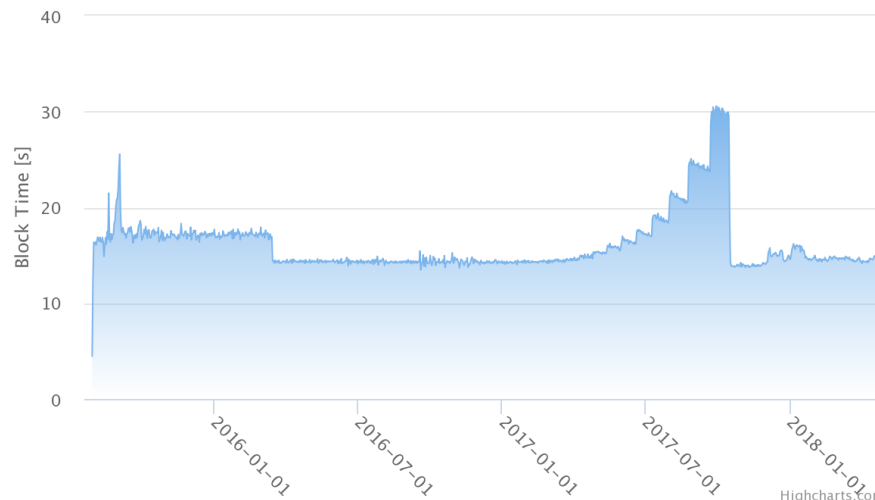
■ [Gas price](#) ~2 gwei

■ [Gas Price](#) / Gas Limit by TX

■ If you run out of gas, state is reverted, ETH gone

Average block time of the Ethereum Network

Source: etherchain.org
Click and drag in the plot area to zoom in



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_{clear}	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 $\lceil \log_{256}(\text{exponent}) \rceil$ 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.

$W_{zero} = \{\text{STOP, RETURN}\}$

$W_{base} = \{\text{ADDRESS, ORIGIN, CALLER, CALLVALUE, CALLDATASIZE, CODESIZE, GASPRICE, COINBASE, TIMESTAMP, NUMBER, DIFFICULTY, GASLIMIT, POP, PC, MSIZE, GAS}\}$

$W_{verylow} = \{\text{ADD, SUB, NOT, LT, GT, SLT, SGT, EQ, ISZERO, AND, OR, XOR, BYTE, CALLDATALOAD, MLOAD, MSTORE, MSTORE8, PUSH*, DUP*, SWAP*}\}$

$W_{low} = \{\text{MUL, DIV, SDIV, MOD, SMOD, SIGNEXTEND}\}$

$W_{mid} = \{\text{ADDMOD, MULMOD, JUMP}\}$

$W_{high} = \{\text{JUMPI}\}$

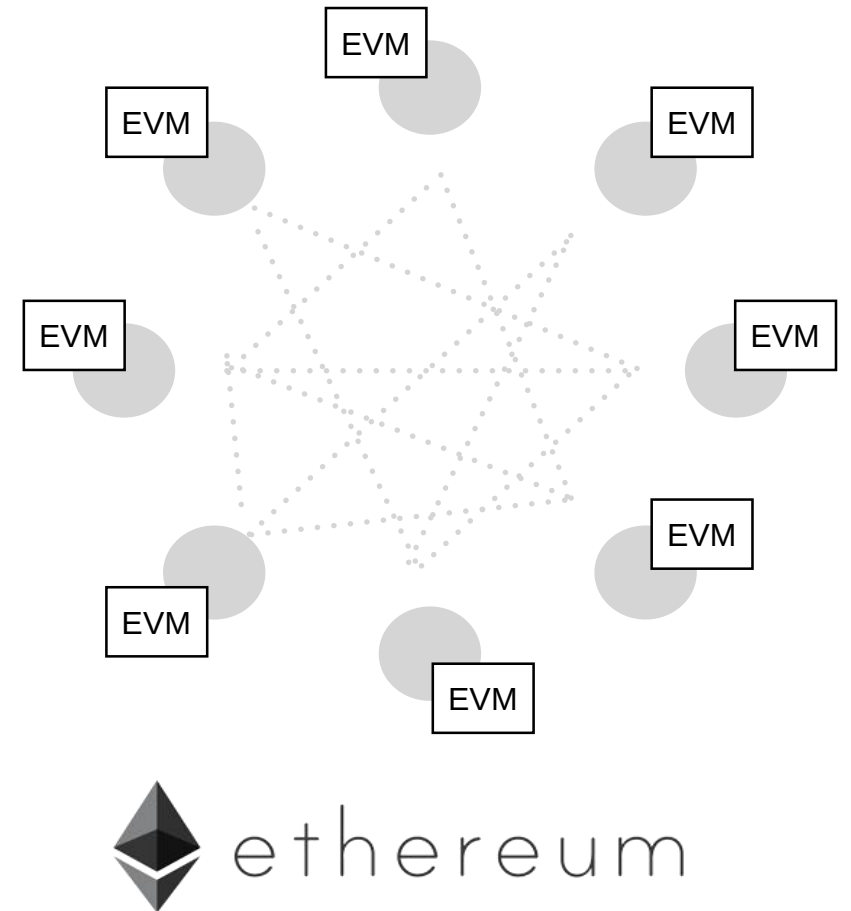
$W_{extcode} = \{\text{EXTCODESIZE}\}$

URL: b.socrative.com/student/

Room: HSR

Ethereum smart contract

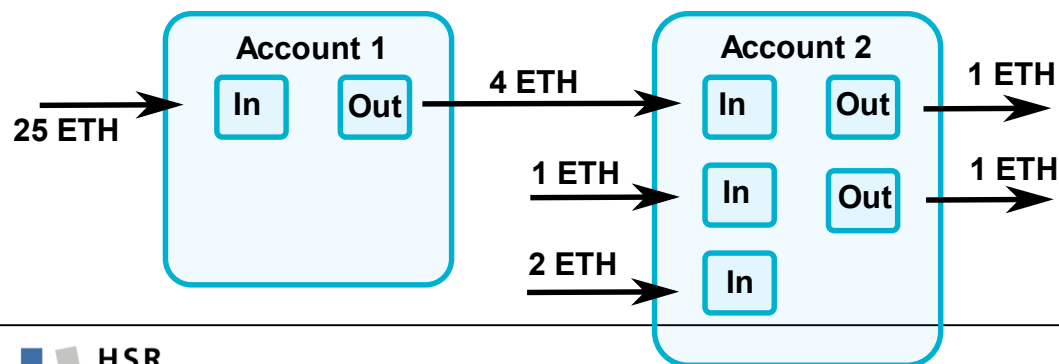
- Computation on [EVM](#) is "very expensive": every contract is run on every full Ethereum node
 - Result on every node is the same
 - Global computer, always running, always correct
- For now, [proof of work \(PoW\)](#)
 - But difficult with ASIC (memory-hard), starts at 1GB RAM for full node and miner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

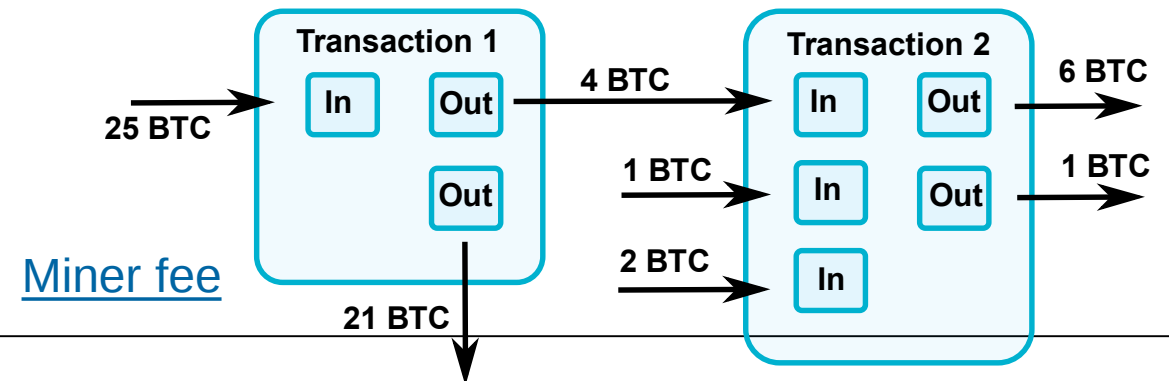
Account-based

- 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

Merkle hash

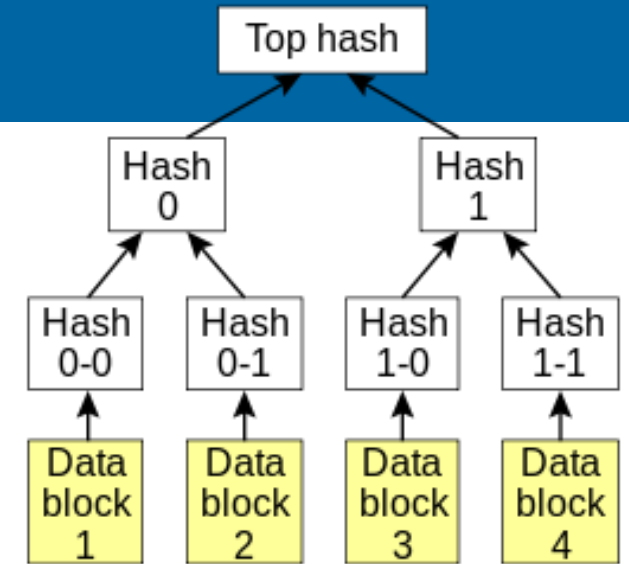
■ Hash Tree

- Tree of hashes ($|| \rightarrow$ concatenation)
- $\text{hash } 0 = \text{hash}(\text{hash } 0-0 || \text{hash } 0-1)$
- $\text{hash } 1 = \text{hash}(\text{hash } 1-0 || \text{hash } 1-1)$
- $\text{Top hash} = \text{hash}(\text{hash } 0 || \text{hash } 1)$

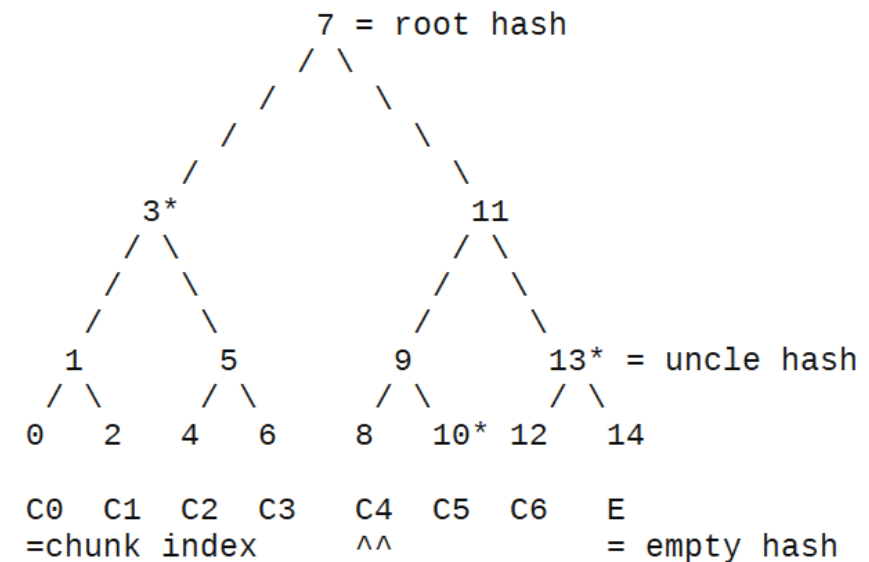
■ Verification (BitTorrent)

- Peer A has top hash (root hash)
 - Peer downloads C4 from peer B
 - create hash 8
 - Need hash 10, 13, 3 (uncle hash)
 - Can be from peer B
 - With 8,10,13,3 can create root hash
- verify this root hash

■ Merkle tree / hash tree also used in [Bitcoin / Ethereum](#)



http://en.wikipedia.org/wiki/Hash_tree



The Merkle hash tree of an interval of width $w=8$

<http://datatracker.ietf.org/doc/draft-ietf-ppsp-peer-protocol/> Section 5.2

URL: b.socrative.com/student/

Room: HSR

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 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)

```
INFO [05-08|17:14:43] Commit new mining work      number=891910 txs=0 uncles=0 elapsed=392.257µs
INFO [05-08|17:15:06] Imported new chain segment  blocks=1 txs=0 mgas=0.000 elapsed=17.225ms mgasps=0.000 number=891910 hash=812c12..de3c2e
INFO [05-08|17:15:06] Commit new mining work      number=891911 txs=2 uncles=0 elapsed=6.039ms
INFO [05-08|17:15:16] Successfully sealed new block number=891911 hash=9efde0..7642c5
INFO [05-08|17:15:16]  ↳ mined potential block     number=891911 hash=9efde0..7642c5
INFO [05-08|17:15:16] Commit new mining work      number=891912 txs=0 uncles=0 elapsed=507.117µs
INFO [05-08|17:15:23] Imported new chain segment  blocks=1 txs=0 mgas=0.000 elapsed=6.596ms mgasps=0.000 number=891912 hash=c80dc0..5fbfde
```

- 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](#)

■ 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

The screenshot displays the Remix IDE interface. At the top, there are tabs for 'Compile', 'Run', 'Settings', 'Analysis', 'Debugger', and 'Support'. The 'Run' tab is active, showing the 'Environment' section with a dropdown menu set to 'JavaScript VM'. Below this, there are fields for 'Account' (0xca3...a733c (99.999999999999738)), 'Gas limit' (3000000), and 'Value' (0). A 'Create' button is visible. Below the 'Create' button, there is a dropdown menu set to 'SecondEpicLuckyCoin' and a 'Load contract from Address' button. At the bottom, there is a section for '0 pending transactions' with a 'Send' button. Below this, there is a section for 'ValidToken at 0x692...77b3a (memory)' with a list of functions and their parameters:

Function	Parameters
approve	address _spender, uint256 _
finishMinting	
lockTokens	address[] _holders, uint256[
mint	address[] _recipients, uint25
transfer	address _to, uint256 _value
transferAndCall	address _to, uint256 _value,
transferFrom	address _from, address _to,
transferOwnership	address _newOwner
allowance	address _owner, address _s
balanceOf	address _owner
decimals	
maxSupply	
mintingDone	
name	
owner	

URL: b.socrative.com/student/

Room: HSR

Questions?

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



Upvote this lecture 😊

1MGDdtoJHQkDEmPwWtwEQxhtRvJfMMAz83