Hands on Ethereum Smart Contracts

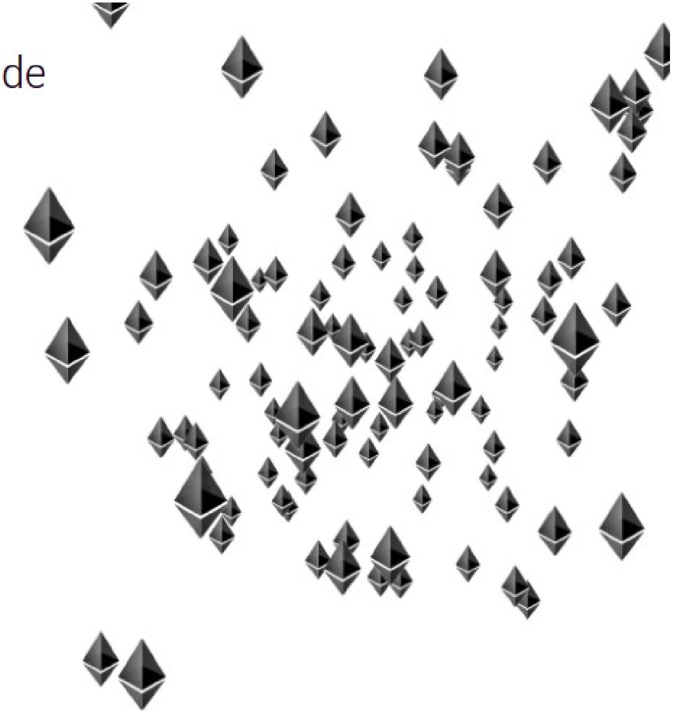
Dan Rusnac

15 Giugno 2019, Macerata

Blockchain

OPEN SOURCE, PUBLIC NETWORK

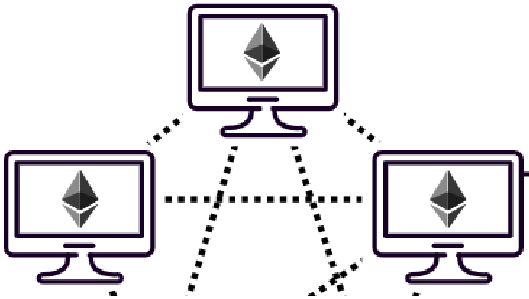
- Anyone can download the software to set up a node
- Permissionless
  - Anyone can join or contribute
- Blockchain features provide:
  - State management
  - Trustlessness
  - Trackability
  - Irreversible
- The Ethereum Virtual Machine
  - Execution environment
- Ideal system for smart contracts

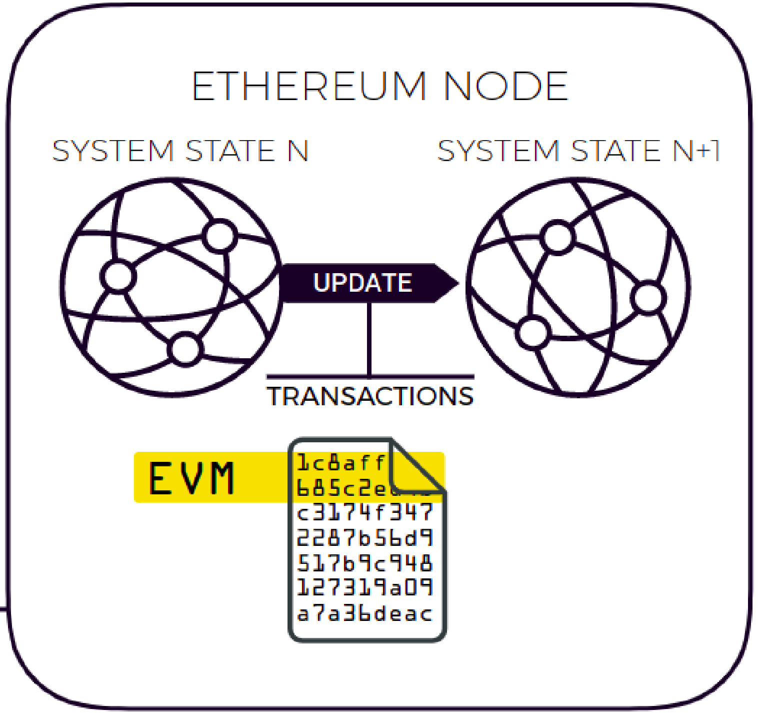


DISTRIBUTED COMPUTING

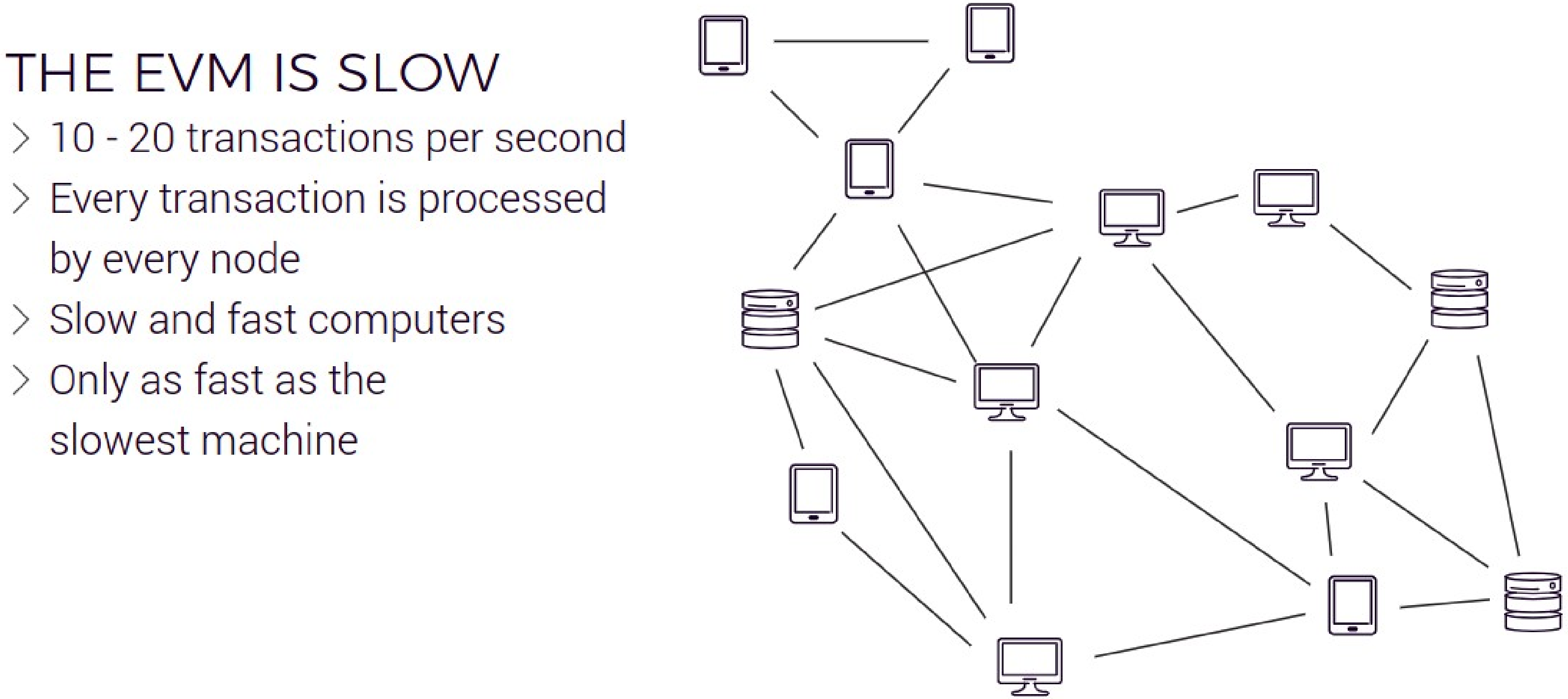
Ethereum Virtual Machine (EVM)

- Runs on every node
- Handles all transaction processing
- Turing complete
- Operates on bytecode





DISTRIBUTED COMPUTING



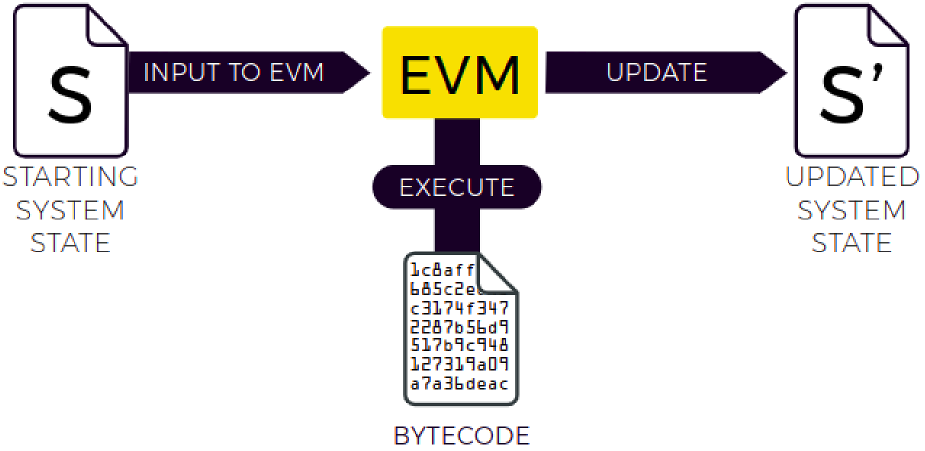
THE EVM IS SLOW

- 10 - 20 transactions per second
- Every transaction is processed by every node
- Slow and fast computers
- Only as fast as the slowest machine

EVM BYTECODE

THE EVM EXECUTES BYTECODE

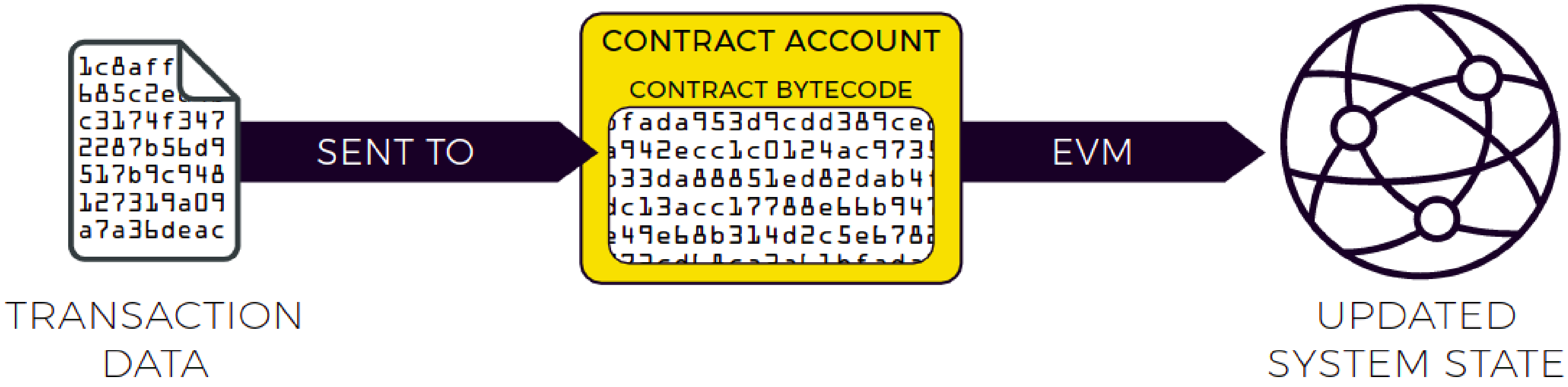
- Bytecode is a low level stack based language
  - Bytecode specifies how state transitions are applied to the network's state
- Transactions contain bytecode



EVM BYTECODE

SMART CONTRACTS EXIST AS BYTECODE

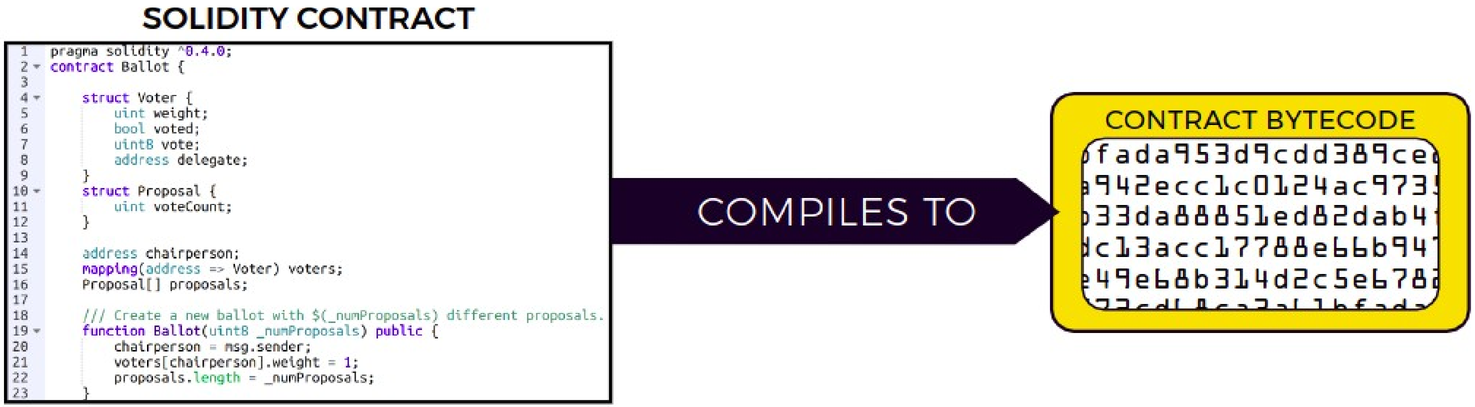
- Contract accounts keep contract bytecode in storage
- EVM executes contract bytecode when a transaction is received



EVM BYTECODE

HIGH LEVEL LANGUAGES COMPILE TO BYTECODE

- Solidity
- Vyper
- LLL

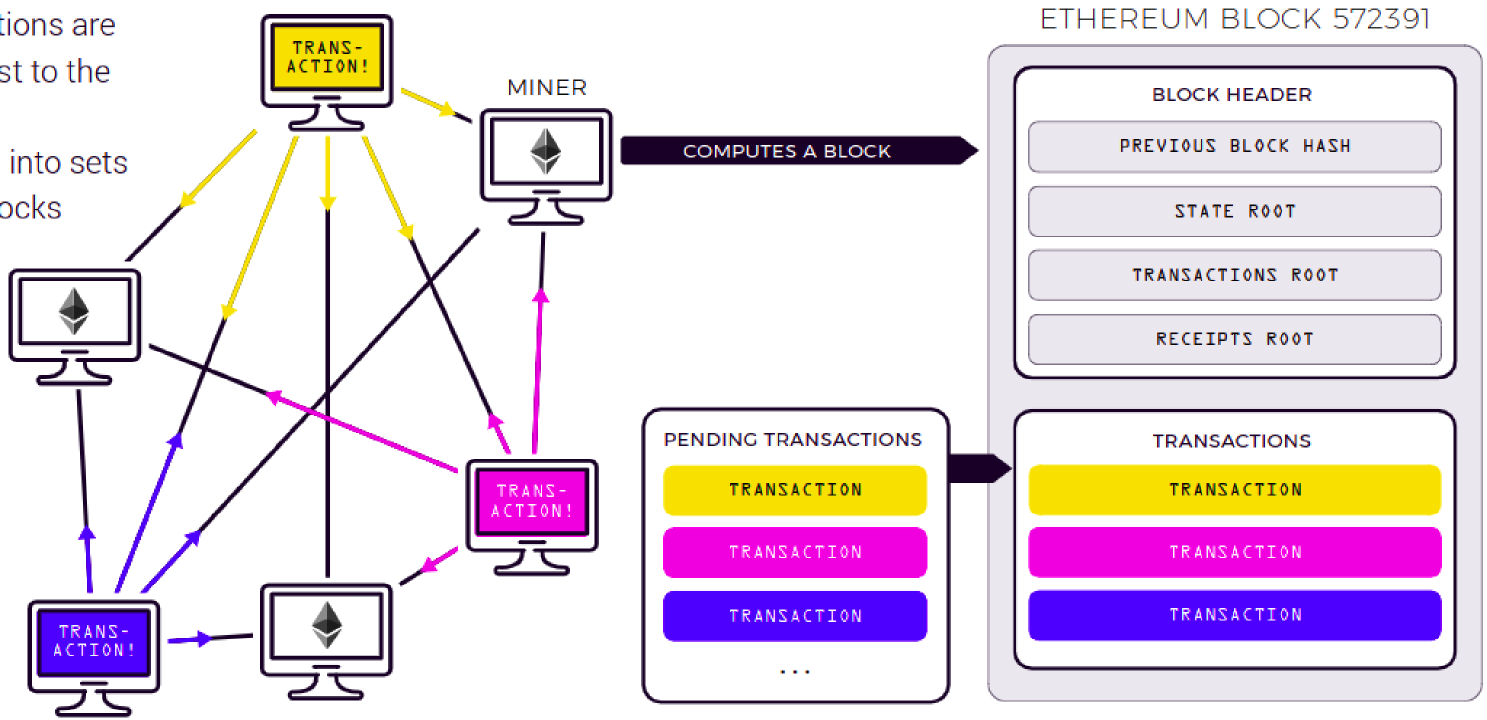


```
pragma solidity ^0.4.0;
contract Ballot {

    struct Voter {
        uint weight;
        bool voted;
        uint8 vote;
        address delegate;
    }
    struct Proposal {
        uint voteCount;
    }

    address chairperson;
    mapping(address => Voter) voters;
    Proposal[] proposals;

    /// Create a new ballot with $(_numProposals) different proposals.
    function Ballot(uint8 _numProposals) public {
        chairperson = msg.sender;
        voters[chairperson].weight = 1;
        proposals.length = _numProposals;
    }
```

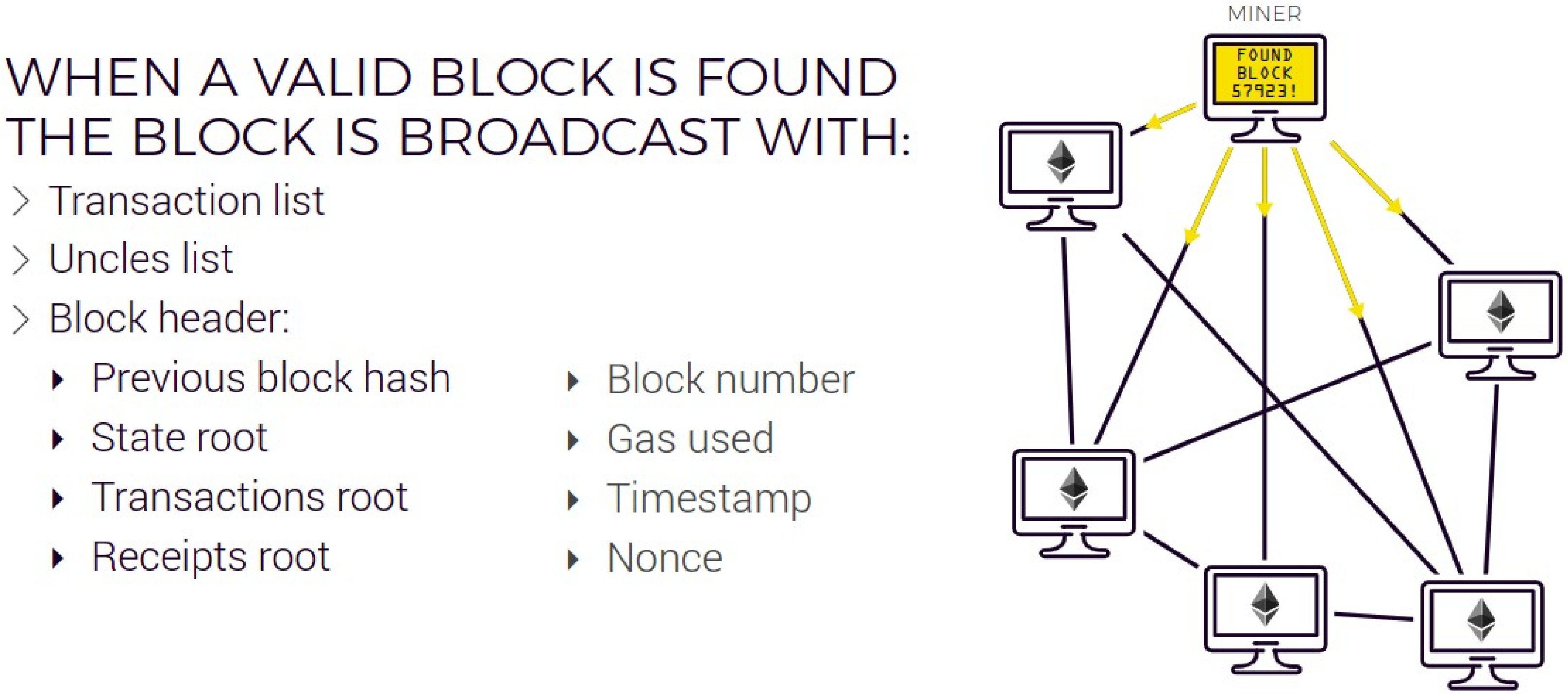
```
fada953d9cdd389ce
942ecc1c0124ac973
33da88851ed82dab4
c13acc17788e66b94
49e68b314d2c5e678
```

ETHEREUM NETWORK PROCESS

- Transactions are broadcast to the network
- Grouped into sets called blocks



ETHEREUM NETWORK PROCESS

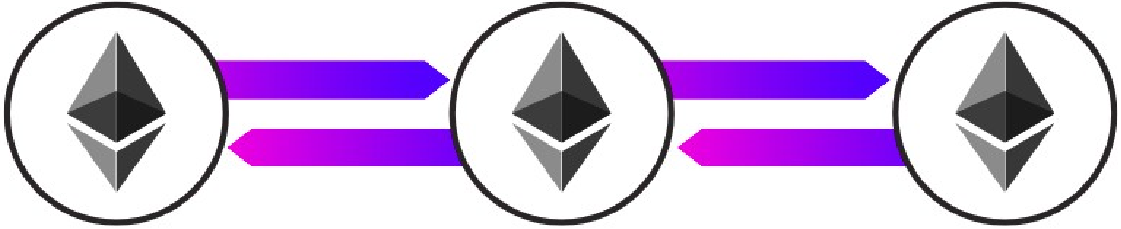


WHEN A VALID BLOCK IS FOUND THE BLOCK IS BROADCAST WITH:

- Transaction list
- Uncles list
- Block header:
  - Previous block hash
  - State root
  - Transactions root
  - Receipts root
  - Block number
  - Gas used
  - Timestamp
  - Nonce

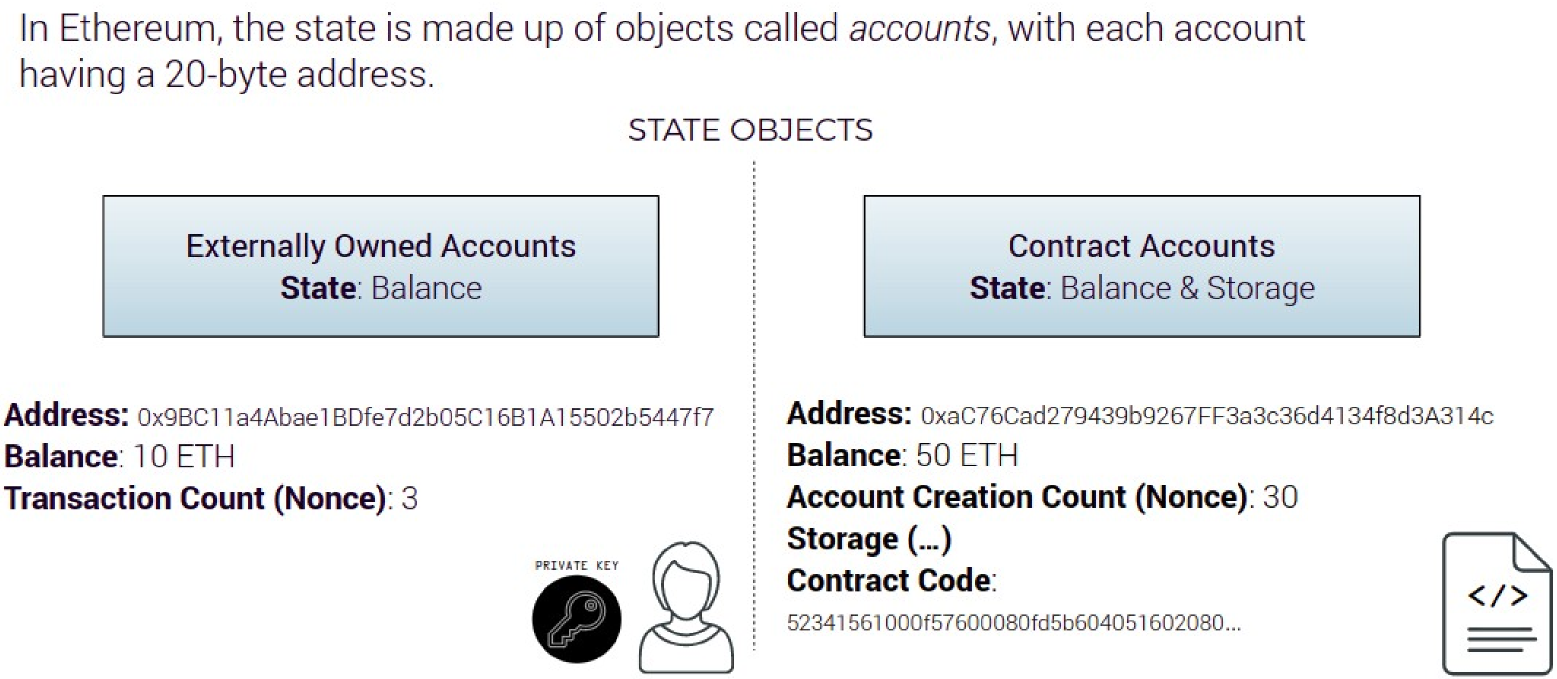
ETHEREUM AS A PLATFORM

- Turing completeness allows for smart contracts
- Digital Identity management
- Value transfer
- Applications on Ethereum
  - Use the underlying Ethereum security
  - Benefit from Ethereum protocol development
- Interoperability
  - Standard protocol



Ethereum basics: accounts

ETHEREUM ACCOUNTS



In Ethereum, the state is made up of objects called *accounts*, with each account having a 20-byte address.

STATE OBJECTS

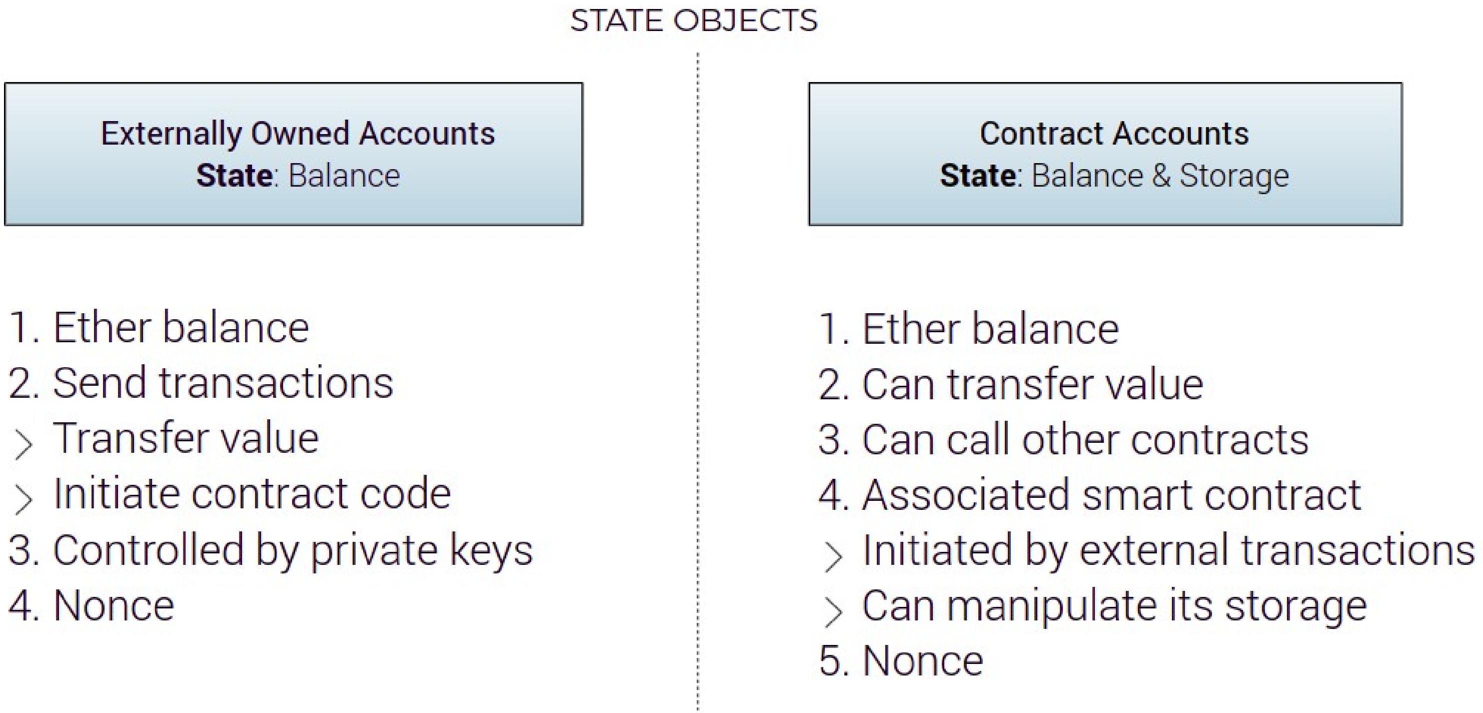
Externally Owned Accounts
State: Balance

Contract Accounts
State: Balance & Storage

Address: 0x9BC11a4Abae1BDfe7d2b05C16B1A15502b5447f7
Balance: 10 ETH
Transaction Count (Nonce): 3

Address: 0xaC76Cad279439b9267FF3a3c36d4134f8d3A314c
Balance: 50 ETH
Account Creation Count (Nonce): 30
Storage (...)
Contract Code:
52341561000f57600080fd5b604051602080...

ETHEREUM ACCOUNTS



STATE OBJECTS

Externally Owned Accounts
State: Balance

1. Ether balance
2. Send transactions
   - › Transfer value
   - › Initiate contract code
3. Controlled by private keys
4. Nonce

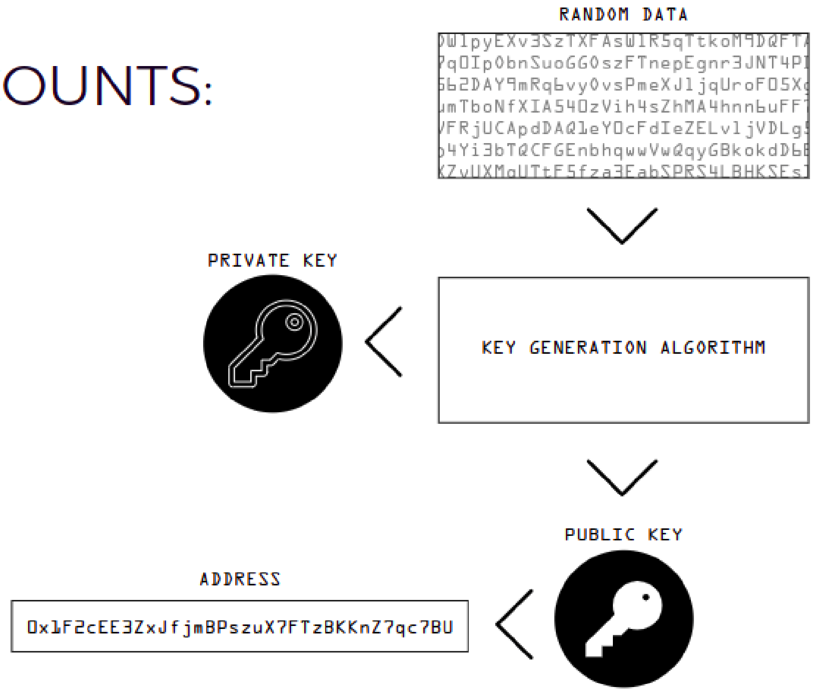
Contract Accounts
State: Balance & Storage

1. Ether balance
2. Can transfer value
3. Can call other contracts
4. Associated smart contract
   - › Initiated by external transactions
   - › Can manipulate its storage
5. Nonce

GENERATING ACCOUNTS

EXTERNALLY OWNED ACCOUNTS:

Whenever a user generates a private-public keypair, a corresponding account address is created as well.



RANDOM DATA

```
WIpyEXv3SzTXFAsWIR5qTtkoM9D@FT
qOIp0bnSuoGG0szFTnepEgnr3JNT4P
62DAY9mRq6vy0vsPmeXJ1jqUroFO5X
mTboNfXIA54OzVih4sZhMA4hnn6uFF
FRjUCApdDA@1eYOcFdIeZELv1jVDLg
4Yi3bT@CFGEnbhqwwVw@qyGBkokdD6
ZvUXMoUTtF5fza3EabSPRS4LBHKSEs
```

KEY GENERATION ALGORITHM

PRIVATE KEY

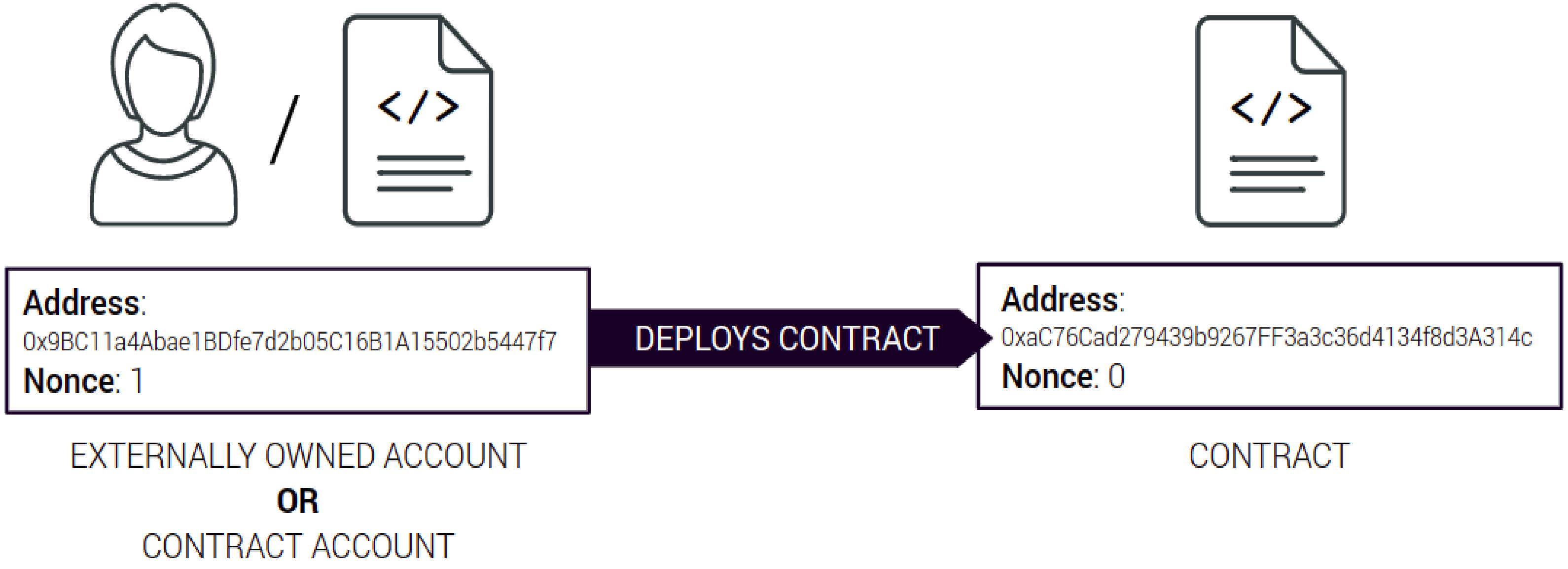
PUBLIC KEY

ADDRESS

0x1F2cEE3ZxJfjmBPszuX7FTzBKKnZ7qc7BU

GENERATING ACCOUNTS

Contract Accounts



Address:
0x9BC11a4Abae1BDfe7d2b05C16B1A15502b5447f7
Nonce: 1

DEPLOYS CONTRACT

Address:
0xaC76Cad279439b9267FF3a3c36d4134f8d3A314c
Nonce: 0

EXTERNALLY OWNED ACCOUNT
OR
CONTRACT ACCOUNT

CONTRACT

Ethereum basics: transactions

ETHEREUM TRANSACTIONS

A MESSAGE SENT FROM ONE ACCOUNT TO ANOTHER

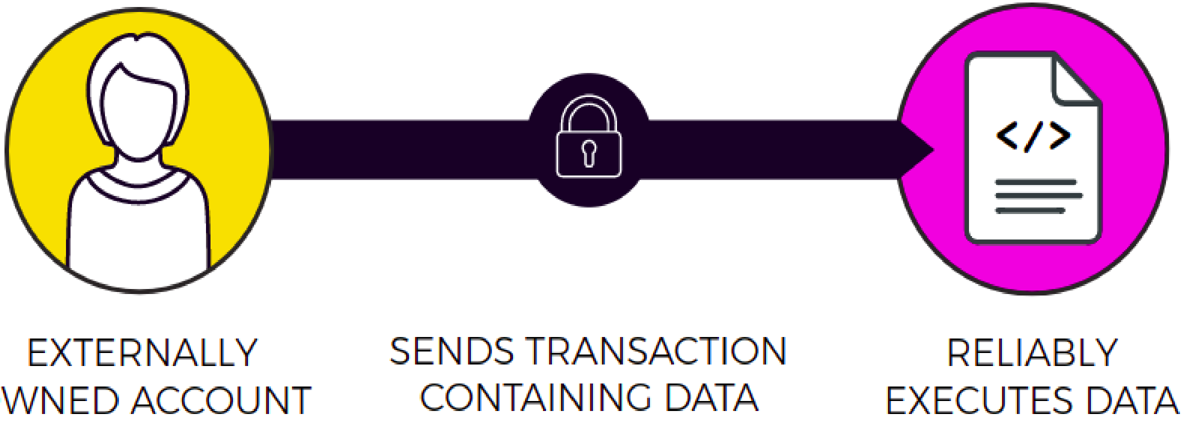
- Transactions update state
  - Ether balance
  - Contract Storage
- Always signed by sender

- EOA or contract account
- Optionally includes
  - Ether
  - Data

ETHEREUM TRANSACTIONS

IF THE TARGET IS A CONTRACT

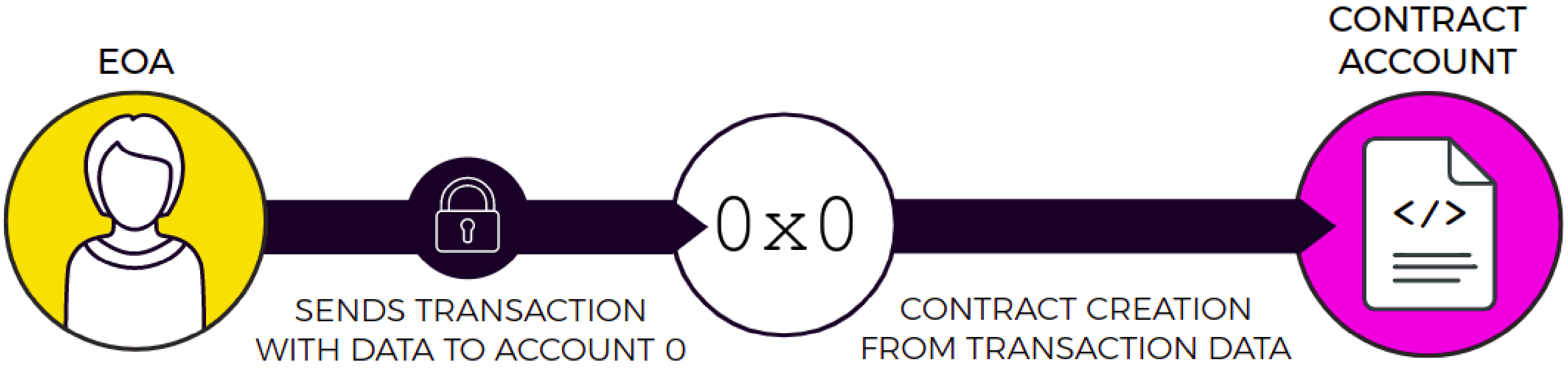
- Code executes with data as input



ETHEREUM TRANSACTIONS

IF TARGET ACCOUNT IS 0

- Transaction creates a new contract
- Transaction data is executed
  - Output is stored as the contract


EOA
SENDS TRANSACTION WITH DATA TO ACCOUNT 0
0x0
CONTRACT CREATION FROM TRANSACTION DATA
CONTRACT ACCOUNT
</ >

ETHEREUM TRANSACTIONS

Ethereum transaction contents

Recipient Address:

`0x9BC11a4Abae1BDfe7d2`
`b05C16B1A15502b5447f7`

Nonce: **`5`**

(Transaction count from sender)

Cryptographic Variables: **V**, **R** and **S**

- Make up the sender's signature

Value (optional): **`100000`** (in wei)

- Amount of Ether to send with the transaction

Data (optional): **`0x8b69a0ca`**

- Specifies contract instructions or deployment instructions

Gas Limit or **Start Gas**

- The maximum number of computational steps the transaction execution is allowed to take

Gas Price

- The fee the sender pays per computational step

Ethereum basics: gas and fees

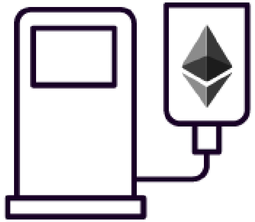
EXECUTION COSTS

EXECUTING OPERATIONS ON ETHEREUM COSTS A FEE

- The network is like a public utility, like the electric grid
- Gas fees are incentives for miners
- Miners collect all gas used in a block as a reward

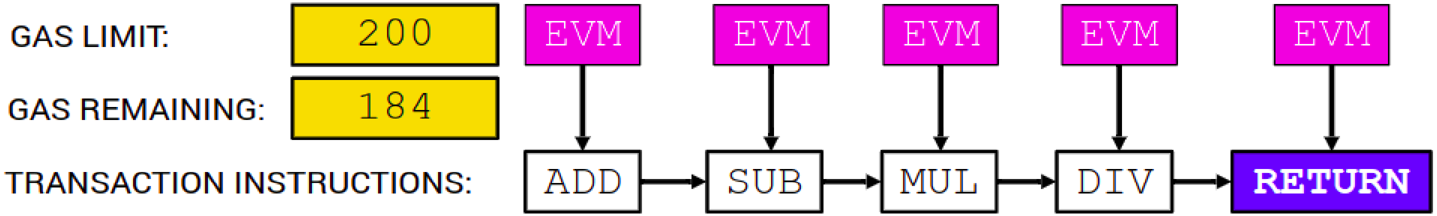
EXECUTION COSTS

- Gas is the metering unit of the Ethereum Virtual Machine
- Each operation on the EVM consumes gas
  - Different operations cost different amounts
  - Multiplication consumes 5 gas
  - Addition consumes 3 gas
- Gas is paid for with Ether
- Gas limit and price is specified per transaction
- Gas limit is max gas allowed for the transaction
- Gas price is how much Ether the sender pays per gas



GAS USAGE

- Every operation consumes a predetermined amount of gas
- Set transaction gas limit at the beginning of a transaction
  - Start gas / gas limit
  - Start gas is sent with transaction to pay for processing
- Every operation consumes gas
  - Remaining gas is reduced every step



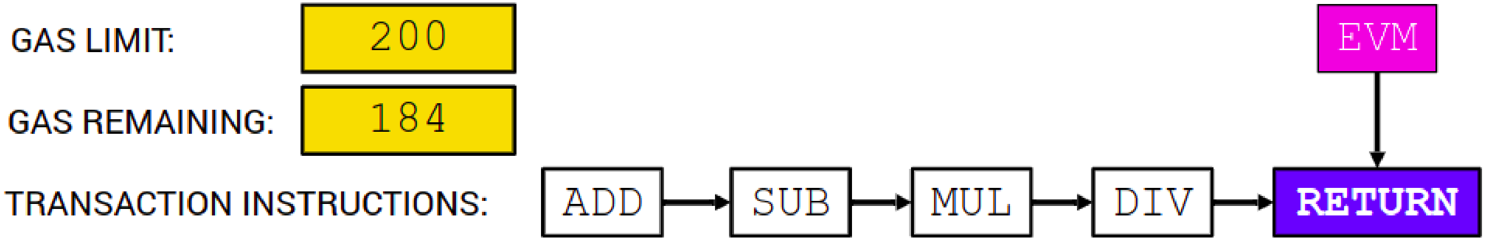
GAS USAGE

OUT OF GAS

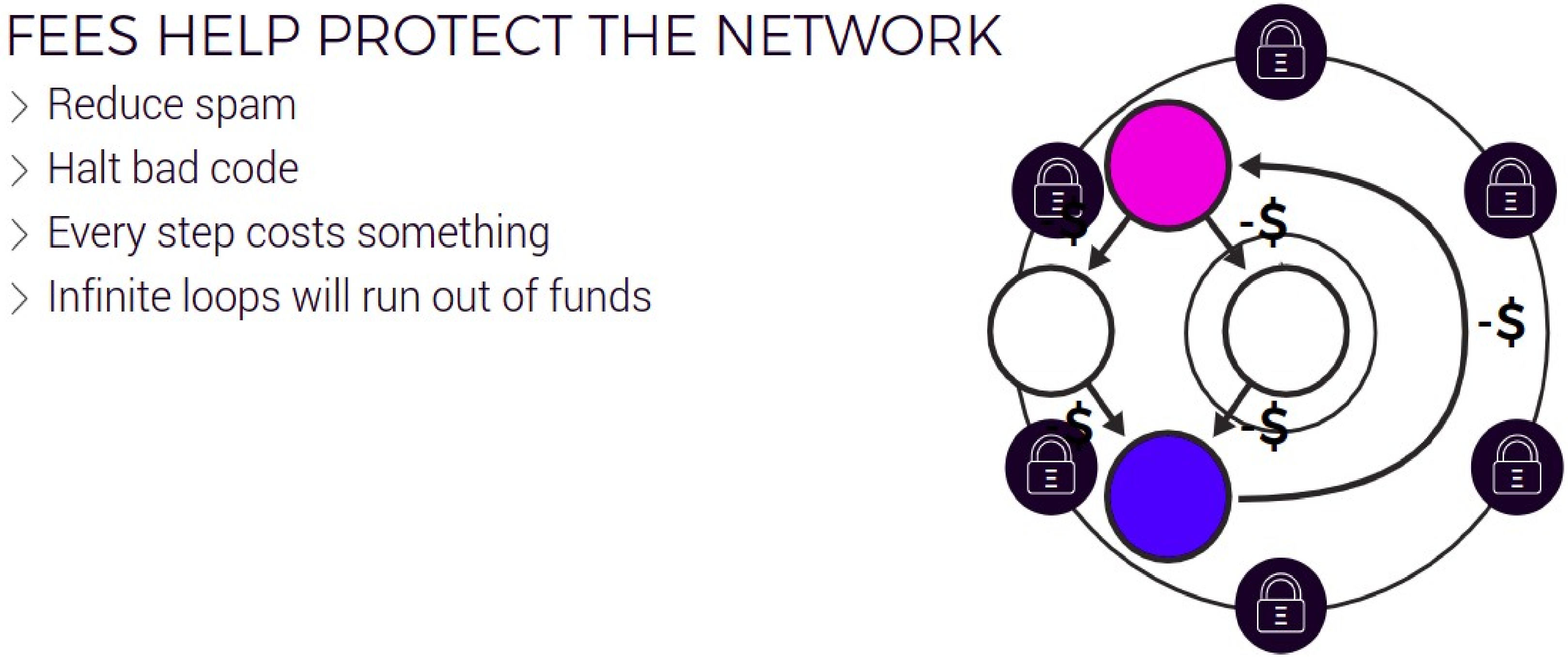
- Transactions that run out of gas with steps remaining will fail

SUCCESSFUL TRANSACTION

- Remaining gas is returned to the sender of the transaction



EXECUTION COSTS

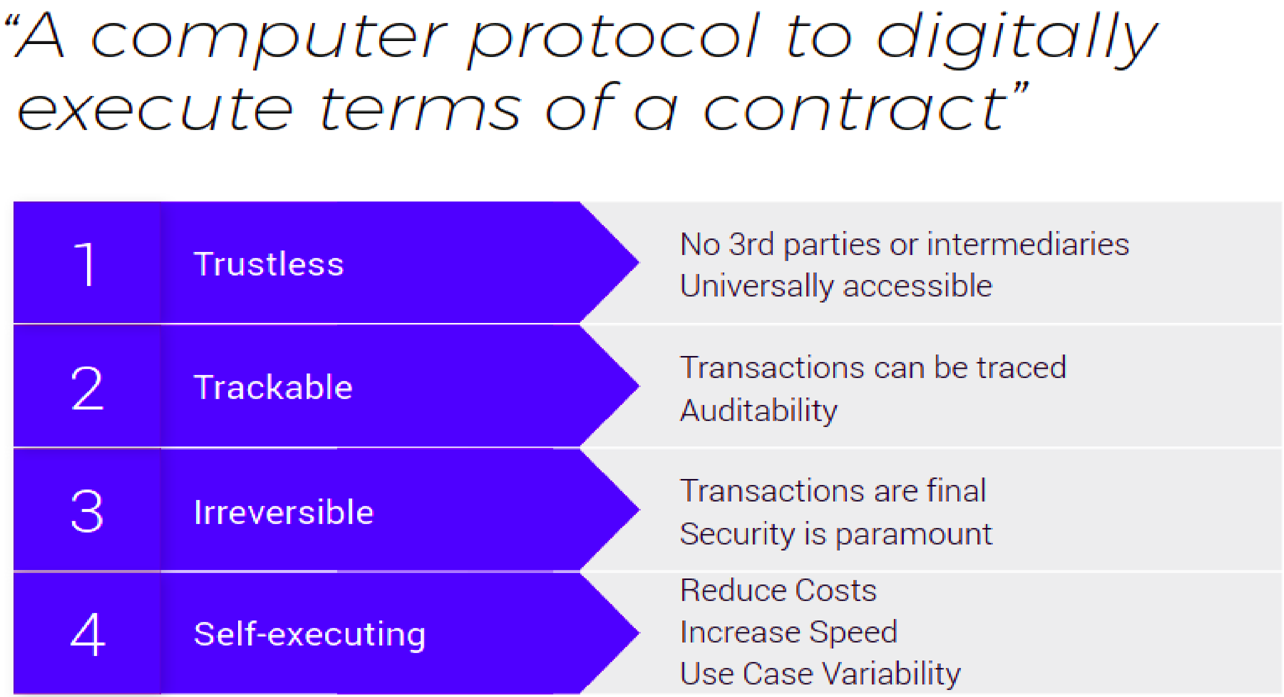


FEES HELP PROTECT THE NETWORK

- Reduce spam
- Halt bad code
- Every step costs something
- Infinite loops will run out of funds

Smart contracts (briefly)

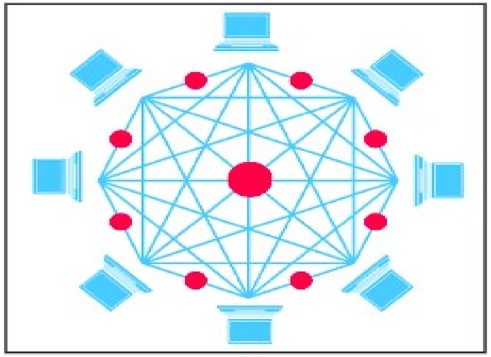
SMART CONTRACTS



“A computer protocol to digitally execute terms of a contract”

1	Trustless	No 3rd parties or intermediaries Universally accessible
2	Trackable	Transactions can be traced Auditability
3	Irreversible	Transactions are final Security is paramount
4	Self-executing	Reduce Costs Increase Speed Use Case Variability

SMART CONTRACTS



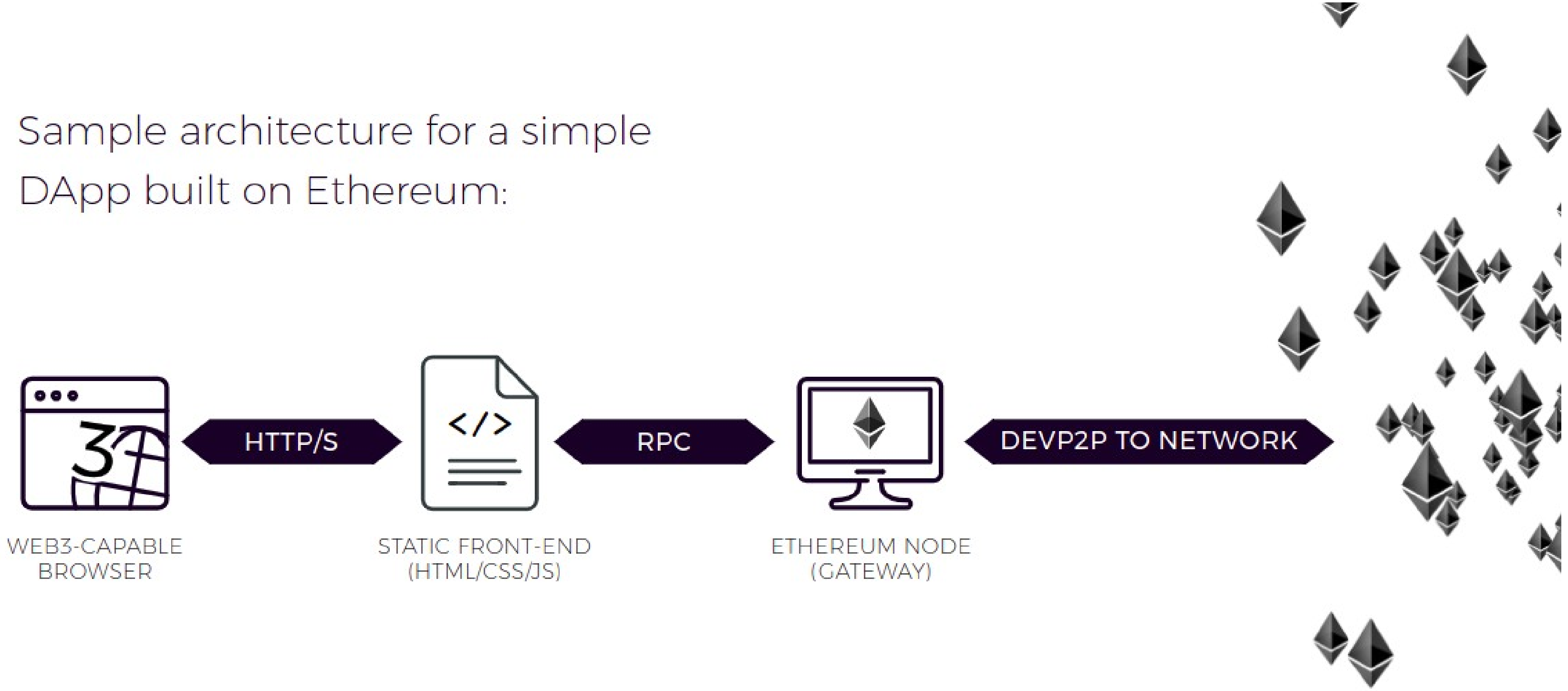
```
a.fn.scrollspy=d,this},a(window).on(                 return this.each(function(){
r),+function(a){"use strict";function b(b){return this.each(function(){var
a[b]()})}var c=function(b){this.element=a(b)};c.VERSION="3.3.7",c.TRANSITIO
opdown-menu)"),d=b.data("target");if(d||(d=b.attr("href"),d=d&&d.replace(/.
it a"),f=a.Event("hide.bs.tab",{relatedTarget:b[0]}),g=a.Event("show.bs.tab"
ultPrevented()){var h=a(d);this.activate(b.closest("li"),c),this.activate(
rigger({type:"shown.bs.tab",relatedTarget:e[0]})})}},c.prototype.activate=
> .active").removeClass("active").end().find('[data-toggle="tab"]').attr(
ia-expanded",!0),h?(b[0].offsetWidth,b.addClass("in")):b.removeClass("fade"
).find('[data-toggle="tab"]').attr("aria-expanded",!0),e&&e()}var g=d.find
e")||!!d.find("> .fade").length);g.length&&h?g.one("bsTransitionEnd",f).em
;var d=a.fn.tab;a.fn.tab=b,a.fn.tab.Constructor=c,a.fn.tab.noConflict=func
show")};a(document).on("click.bs.tab.data-api",'[data-toggle="tab"]',e).on
se strict";function b(b){return this.each(function(){var d=a(this),e=d.dat
typeof b&&e[b]()})}var c=function(b,d){this.options=a.extend({},c.DEFAULT
",a.proxy(this.checkPosition,this)).on("click.bs.affix.data-api",a.proxy(
null,this.pinnedOffset=null,this.checkPosition()};c.VERSION="3.3.7",c.RES
State=function(a,b,c,d){var e=this.$target.scrollTop(),f=this.$element.o
"bottom"==this.affixed)return null!=c?!(e+this.unpin<=f.top)&&"bottom":!(
!=c&&e<=c?"top":null!=d&&i+j>=a-d&&"bottom"},c.prototype.getPinnedOffset
.RESET).addClass("affix");var a=this.$target.scrollTop(),b=this.$element
dthEventLoop=function(){setTimeout(a.proxy(this.checkPosition,this),1)}
et.height(),d=this.options.offset,e=d.top,f=d.bottom
eof e&&(e=d.top(this.$element)),
nt.css("top",
```

Smart Contracts do not necessarily have to be on a blockchain, but blockchains offer:

- A trustless, universally accessible system
- Trackable, irreversible transactions
- Mechanisms for self-execution

Development environment and tools

ANATOMY OF A DECENTRALIZED APPLICATION



Sample architecture for a simple DApp built on Ethereum:

DEVELOPMENT BLOCKCHAIN OPTIONS

ETHEREUM NETWORKS

ETHEREUM MAINNET

⍰ Highest security
⍰ Real value
⍰ Immutable
⍰ Public

⍰ Expensive
⍰ Slow
⍰ Highest risk

TESTNETS

⍰ Public
⍰ Free ether

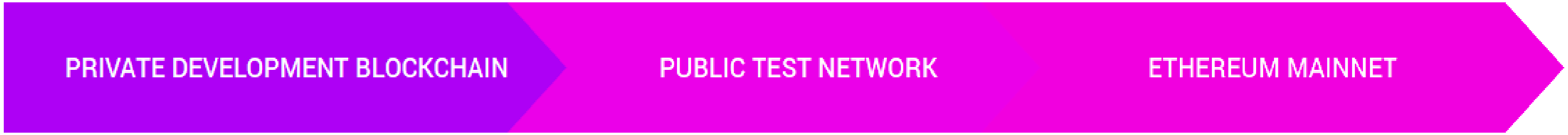
⍰ Not final
⍰ Still slow

PRIVATE NETWORK

⍰ Fast
⍰ Free
⍰ Private

⍰ Isolated
⍰ No Real Value

DEVELOPMENT BLOCKCHAIN PROGRESSION



PRIVATE DEVELOPMENT BLOCKCHAIN

- Great for early development
- Rapid setup, deployment and testing
- Cheap

PUBLIC TEST NETWORK

- Open to the public
- Closer to production environment
- Test larger scale
- Bug bounties
- Slower iterations

ETHEREUM MAINNET

- Highest risk
- Using real value

CONNECTING TO A BLOCKCHAIN

Geth	→	public / private blockchains
Ganache	→	local private development blockchain
Remix	→	blockchain simulator (local, private)
Metamask	→	public / private blockchains

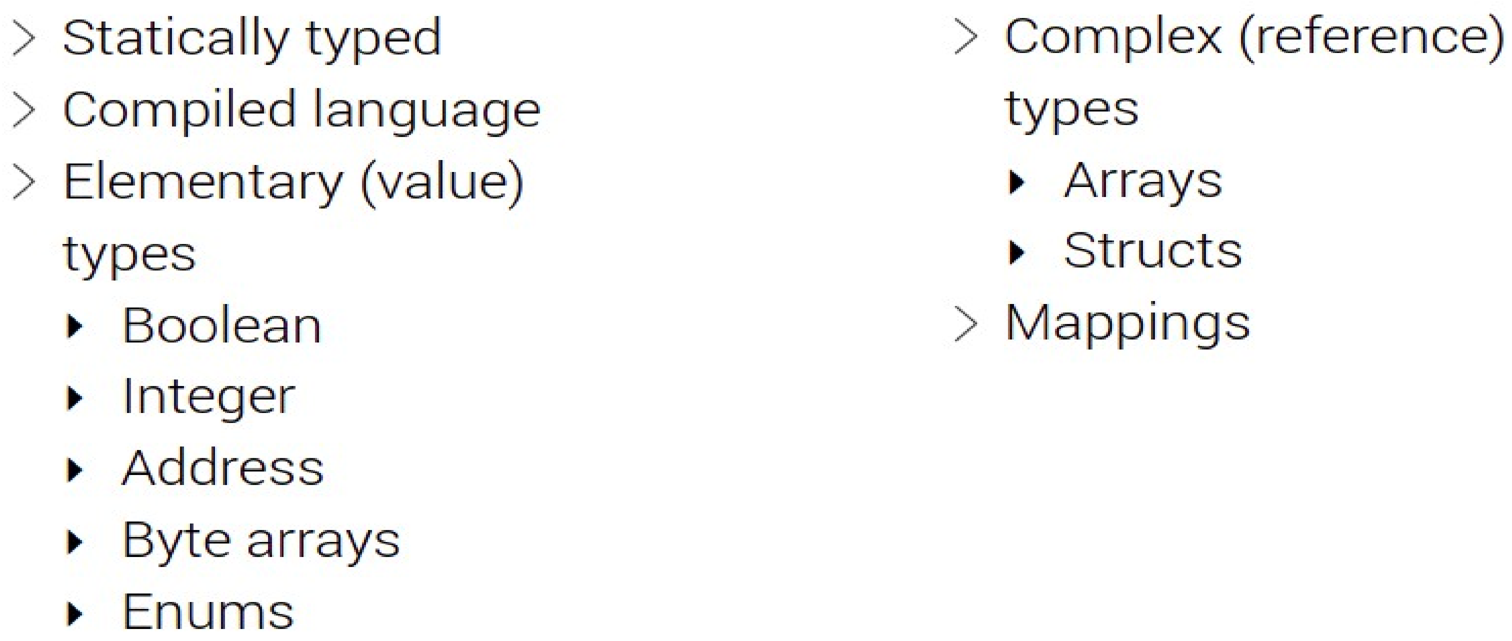
WHAT WE’LL USE

- Remix
- Web3.js
- Npm (Node.js)
- Metamask

Let’s start (simple)!

Smart contracts fundamentals: data types and variables

FEATURES OF SOLIDITY



- Statically typed
- Compiled language
- Elementary (value) types
  - Boolean
  - Integer
  - Address
  - Byte arrays
  - Enums
- Complex (reference) types
  - Arrays
  - Structs
- Mappings

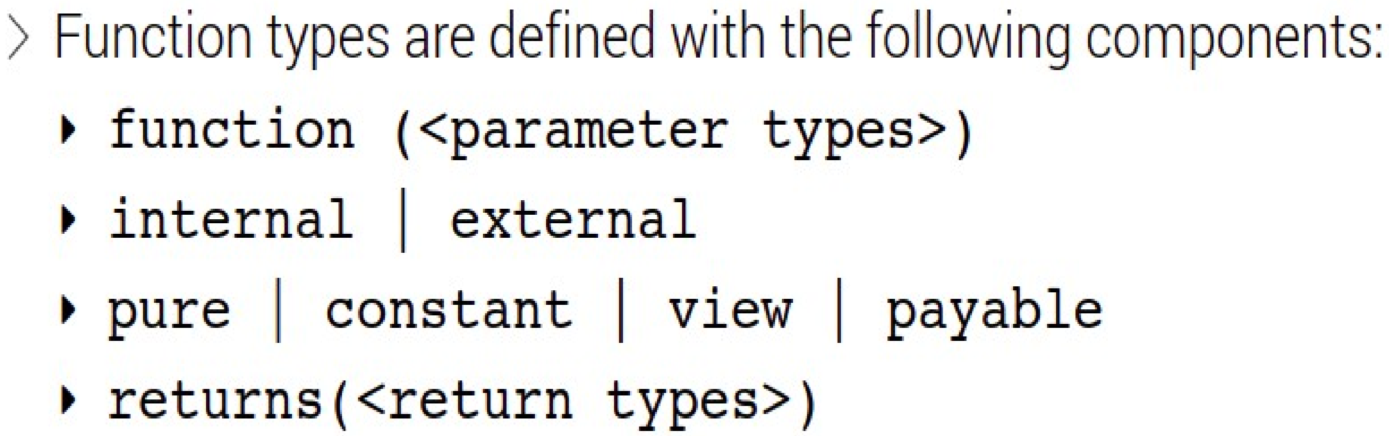
BOOLEAN

- Declared with `bool <variable name>`
- Can be either true or false
- Use logical operators to control flow in conditional statements
- Logical negation (`!`), logical conjunction (`&&`), logical disjunction (||), equality (==) and inequality (`!=`)
- Common short circuiting rules
- Adding the public attribute, `bool public <variable name>`, will automatically create a getter function to retrieve the variables value
  - This is true of all variables
- All types initialize to `0`; unassigned booleans default to `false`.

INTEGER

- Signed or unsigned integers (`int` and `uint`)
- Can be defined with or without assignment (default assignment = 0)
- Can be defined with or without a number suffix
- `uint` is the same as `uint256` where 256 is the size of the integer in bits
- Suffix must be a multiple of 8, `uint8`, `int16`, `int64`, `uint256` are all valid

FUNCTION TYPES



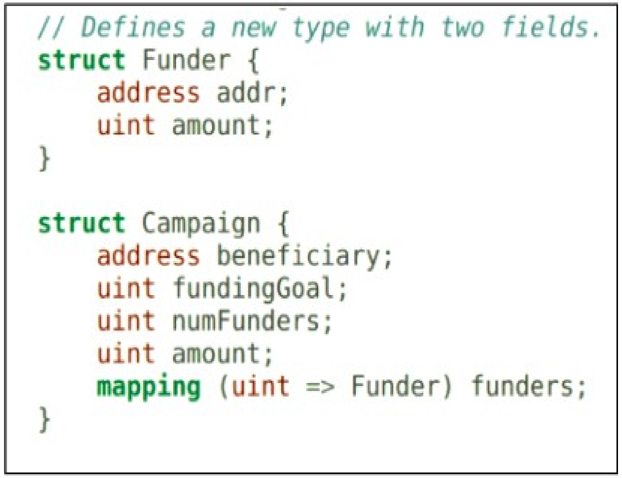
- Function types are defined with the following components:
  - `function (<parameter types>)`
  - `internal | external`
  - `pure | constant | view | payable`
  - `returns(<return types>)`

ARRAYS

- Defined as `T[k]` for fixed size of length `k`
- `T[]` for dynamic
- Can be allocated to storage or memory
  - Storage arrays can be any data type
  - Memory arrays can be anything but a mapping
- Declaring the variable public will create a getter function that requires the index of the desired value as a parameter

STRUCTS

- A way to define new types
- Cannot contain a member of its own type
- Struct values stored as local variables in functions are not copied, they are passed by reference



```
// Defines a new type with two fields.
struct Funder {
    address addr;
    uint amount;
}

struct Campaign {
    address beneficiary;
    uint fundingGoal;
    uint numFunders;
    uint amount;
    mapping (uint => Funder) funders;
}
```

MAPPINGS

- Declared by `mapping(_KeyType => _ValueType)`
- `_Keytype` can be any type but mapping, dynamic array, contract, enum or struct
- `_Valuetype` can be any type
- Mappings are essentially hash tables
  - Every value is virtually initialized to its default
    - As a result, mappings have no length
  - Key data is not stored in the mapping, rather its keccack256 hash
- A mapping declared public will create a getter requiring the `_keyType` as a parameter and return the `_valuetype`

UNITS

- Solidity contains simple unit conversions and globally accessible variables
- Ether units
  - Solidity will convert amounts of ether
  - It recognizes the units of **`wei`**, **`finney`**, **`szabo`** and **`ether`**
- Time suffixes
  - **`1 == 1 seconds`**
  - **`1 minutes == 60 seconds`**
  - **`1 hours == 60 minutes`**
  - **`1 days == 24 hours`**
  - **`1 weeks == 7 days`**
  - **`1 years == 365 days`**
- Leap seconds are not taken into account

GLOBAL VARIABLES

- `msg.data (bytes)`: complete calldata
- `msg.gas (uint)`: remaining gas
- `msg.sender (address)`: sender of the message (current call)
- `msg.sig (bytes4)`: first four bytes of the calldata (i.e. function identifier)
- `msg.value (uint)`: number of wei sent with the message
- `now (uint)`: current block timestamp (alias for `block.timestamp`)
- `tx.gasprice (uint)`: gas price of the transaction
- `tx.origin (address)`: sender of the transaction (full call chain)

Let’s do something bigger!

What now?

Suggestions

- Cryptozombies https://cryptozombies.io/
- Mastering Ethereum https://github.com/ethereumbook/ethereumbook
- Fork my code and let’s build together!

Dan Rusnac

www.danrusnac.ml

Grazie!