

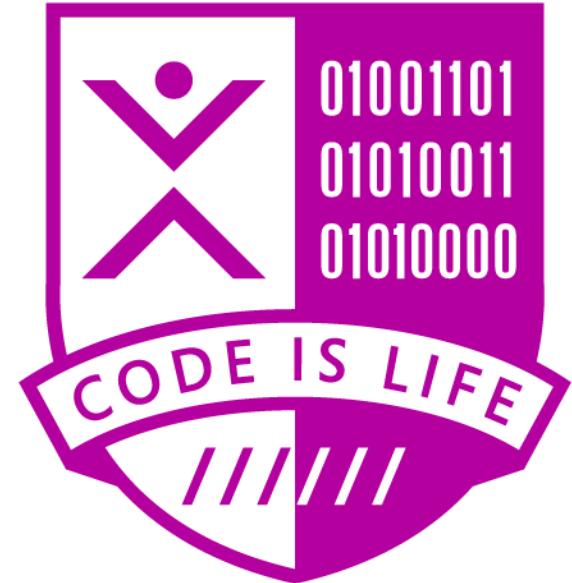
# Microsoft Student Partners

Microsoft Student Partners are the future of technology!

# Applied Ethereum: Blockchain & Smart Contracts



**StudentPartners.de**



[http://blogs.msdn.microsoft.com/  
StudentPartner](http://blogs.msdn.microsoft.com/StudentPartner)

# Agenda

01001101  
01010011  
01010000

# Agenda

Time	Agenda Items
	General Introduction to the Blockchain
	Setup Infrastructure
	Ethereum & Smart Contracts
	Smart Contract Use Cases
	Build your own Smart Contract!
	Present your contract to your peers!

01001101  
01010011  
01010000

# Files for this workshop

Please visit:

<https://goo.gl/kjNdsc>

# Blockchain

01001101  
01010011  
01010000

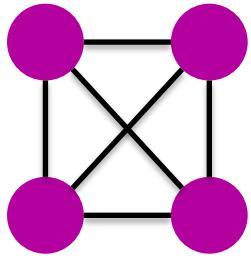
# What is a Blockchain?

- Transnational Database
  - But: **Everyone** can read entries
- Append-only data structure
- **Peer-to-Peer** network
- Transactions stored in **Blocks**

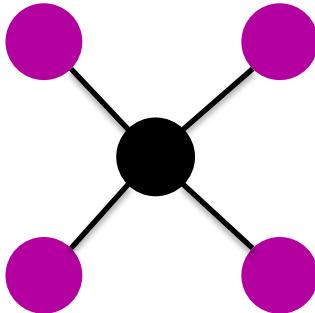


01001101  
01010011  
01010000

# Peer-to-peer vs. centralized architecture



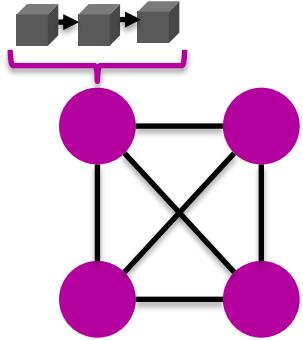
Peer-to-peer  
network



Centralized  
network

01001101  
01010011  
01010000

# Peer-to-peer vs. centralized architecture



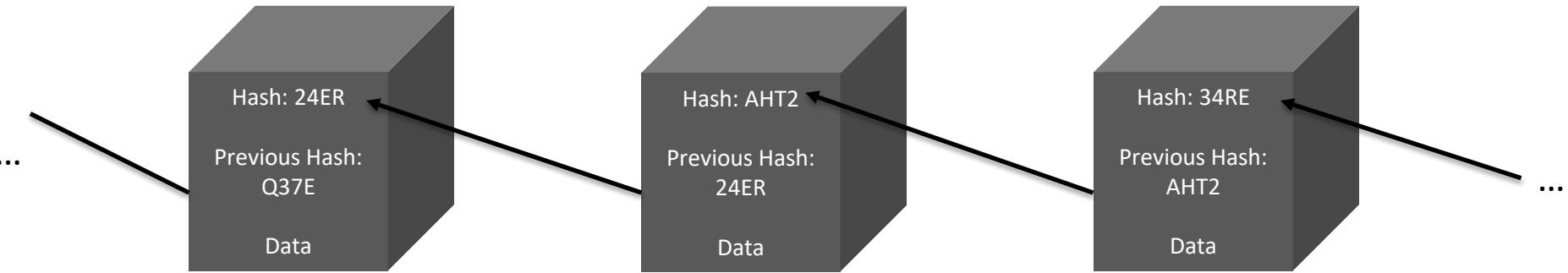
Peer-to-peer  
network

- Each (full) node stores the whole transactional database (blockchain)

01001101  
01010011  
01010000

# Blocks

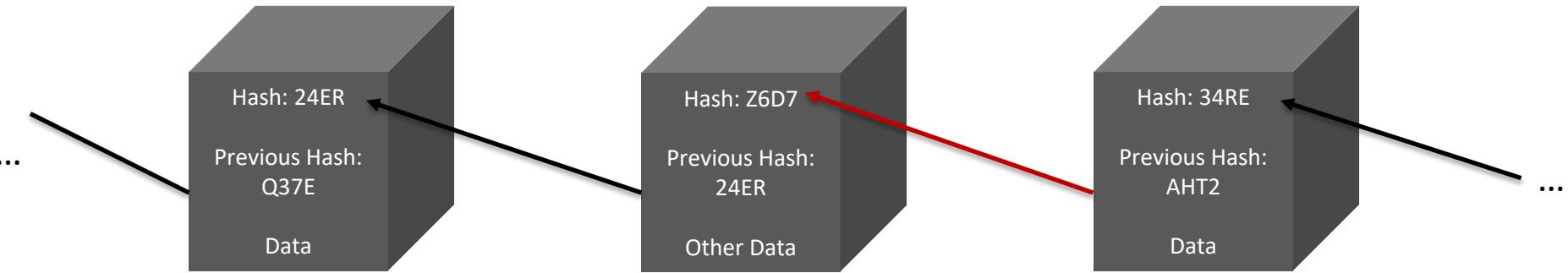
What happens if one block is manipulated?



01001101  
01010011  
01010000

# Blocks

What happens if one block is manipulated?



01001101  
01010011  
01010000

# Full Node vs. Light Node

- Full Node:
  - Stores full blockchain (Bitcoin blockchain: 158.7 Gb)
  - Is connected to other full nodes
    - Miner: Additionally able to create new blocks
- Light Node:
  - Only validates single transactions

01001101  
01010011  
01010000

# Adding new transactions

1. Wallet owner creates and signs transaction
2. Transaction is sent to a full node
3. Transaction is spread through the network
4. Miner builds transaction into block
5. Other nodes validate block and add it to the blockchain

01001101  
01010011  
01010000

# Blockchain Use Cases



steemit



ethereum

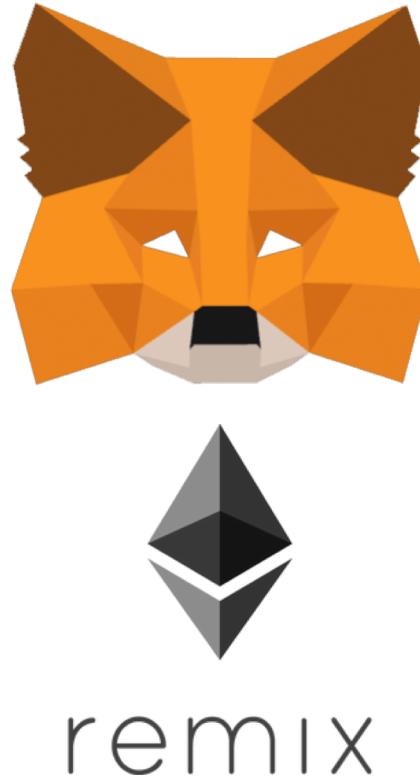
# Required Infrastructure

01001101  
01010011  
01010000

# Getting started

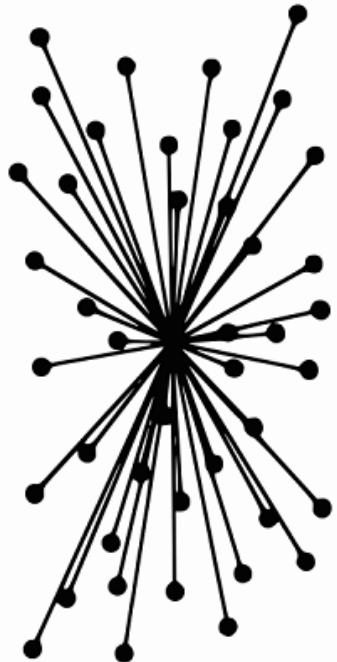
## What are we using?

- Metamask Ethereum browser extension
- Remix IDE
- Ropsten Test Network

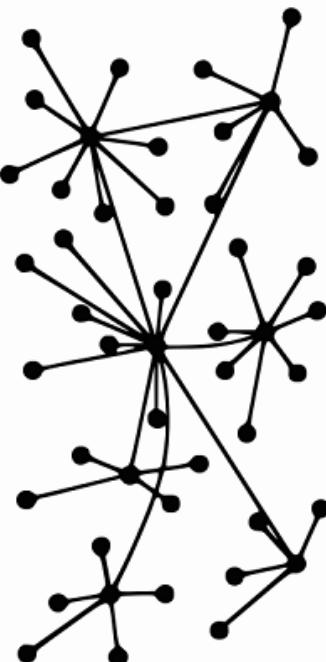


01001101  
01010011  
01010000

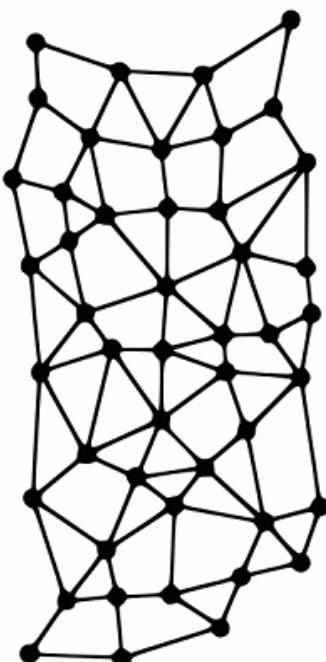
# Centralized vs Decentralized



Centralized



Decentralized



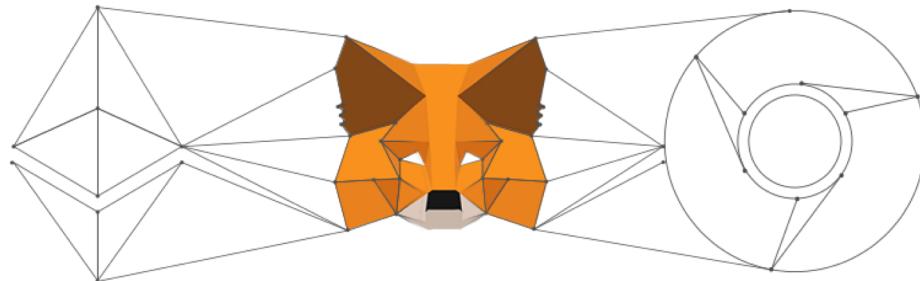
Distributed

01001101  
01010011  
01010000

# Metamask

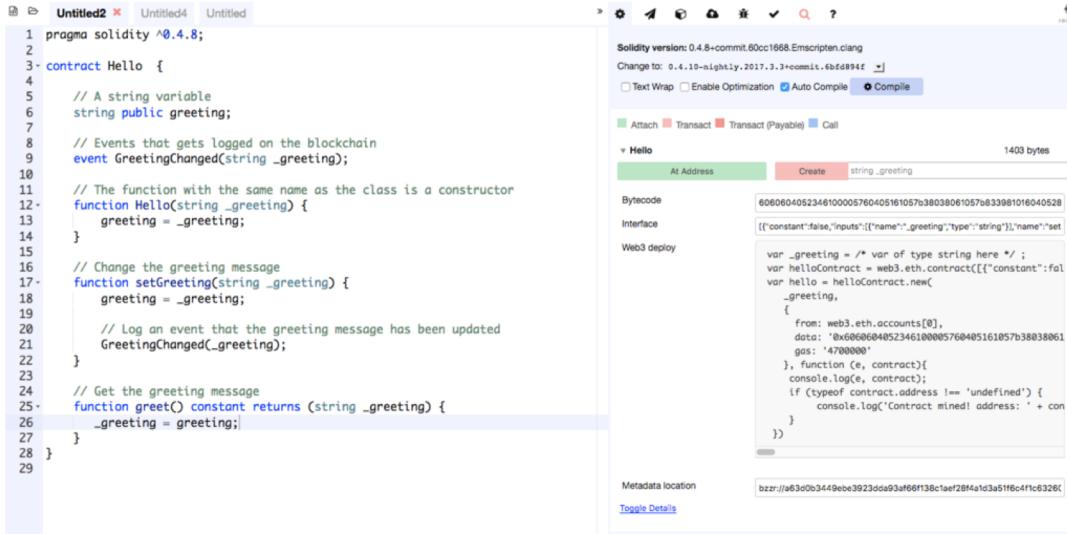
## Why we're using Metamask?

1. In order to write on the Ethereum blockchain we need an Ethereum Node
2. Rather than installing the node on the computer, we're emulating the Ethereum Node in our web browser



# Remix

- Development environment for the browser
- Easiest way to get started with development



The screenshot shows the Remix IDE interface. On the left is a code editor with tabs for Untitled2, Untitled4, and Untitled. The code editor contains the following Solidity code:

```
pragma solidity ^0.4.8;
contract Hello {
    // A string variable
    string public greeting;
    // Events that gets logged on the blockchain
    event GreetingChanged(string _greeting);
    // The function with the same name as the class is a constructor
    function Hello(string _greeting) {
        greeting = _greeting;
    }
    // Change the greeting message
    function setGreeting(string _greeting) {
        greeting = _greeting;
    }
    // Log an event that the greeting message has been updated
    event GreetingChanged(string _greeting);
    // Get the greeting message
    function greet() constant returns (string _greeting) {
        _greeting = greeting;
    }
}
```

On the right side of the interface, there is a sidebar with the following sections:

- Solidity version: 0.4.8+commit.60cc1668.Emscripten.clang
- Change to: 0.4.10-nightly.2017.3.3+commit.6bd6894f
- Text Wrap, Enable Optimization, Auto Compile, Compile (selected)
- Attach, Transact, Transact (Payable), Call
- Hello
- At Address, Create (selected), string \_greeting
- Bytecode: 6060604052346100005760405161057b38038061057b833981016040528
- Interface: [{"constant":false,"inputs":[{"name":"\_greeting","type":"string"}],"name":"setGreeting","outputs":[],"stateMutability":"nonpayable","type":"function"}]
- Web3 deploy
- var \_greeting = /\* var of type string here \*/;  
var helloContract = web3.eth.contract([{"constant":false,"inputs":[{"name":"\_greeting","type":"string"}],"name":"setGreeting","outputs":[],"stateMutability":"nonpayable","type":"function"}]).at('0x6060604052346100005760405161057b38038061057b833981016040528');  
var hello = helloContract.new(\_greeting,{  
 from: web3.eth.accounts[0],  
 data: '0x6060604052346100005760405161057b38038061057b833981016040528',  
 gas: '4700000'  
}, function(e, contract){  
 console.log(e, contract);  
 if (typeof contract.address !== 'undefined') {  
 console.log('Contract mined! address: ' + contract.address)  
 }  
});
- Metadata location: bzz://a63d0b3449ebe3923dd93af68f138c1aef28f4a1d3a5f16c4f1c6326
- Toggle Details



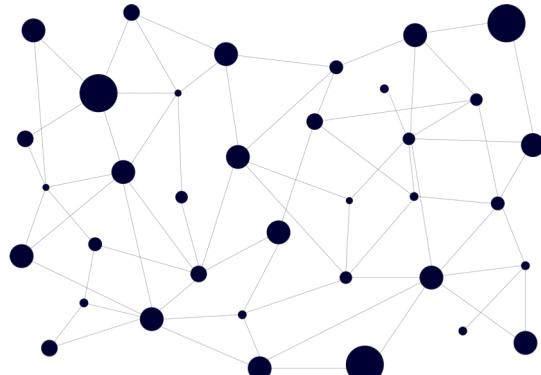
remix

01001101  
01010011  
01010000

# Ropsten

Why using a Test Network and not the Main network?

1. Deploy buggy code to the test network rather than the main network
2. Most up-to-date network
3. It's **free**

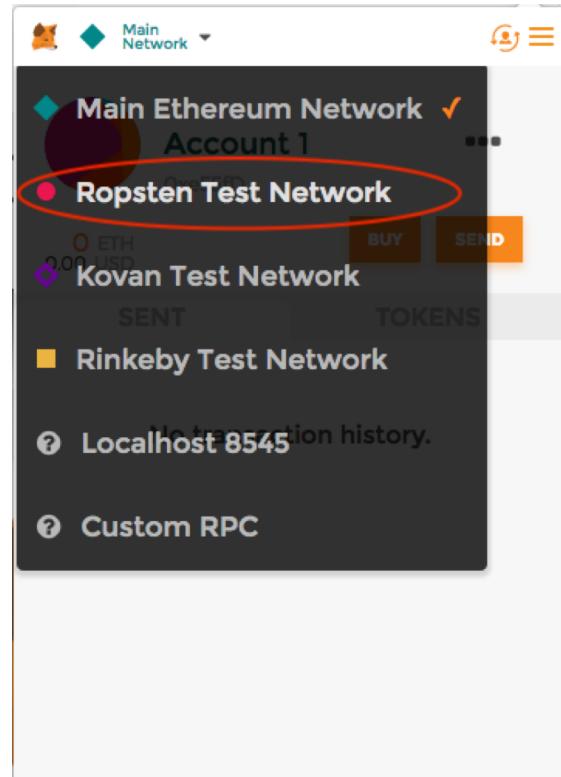


# Setup

01001101  
01010011  
01010000

# Setup – Metamask

1. Head to <https://metamask.io/>
2. Install the Browser Addon
3. Setup a password and save the passphrase
4. Switch to the Ropsten Test Network



01001101  
01010011  
01010000

# Setup Metamask

6. Head to <https://faucet.metamask.io/>

7. Request Ether for your Metamask wallet



## MetaMask Ether Faucet

### faucet

address: 0x81b7e08f65bd5648606c89998a9cc8164397647  
balance: 4695048.36 ether

[request 1 ether from faucet](#)

### user

address: 0xe55fd3e1f6ce574ac790aa7afdf5e355ae2bfe0c5  
balance: 4.00 ether  
donate to faucet:

[1 ether](#) [10 ether](#) [100 ether](#)

### transactions

01001101  
01010011  
01010000

# Etherscan

- Tool to see all transactions you've made on the Ethereum blockchain
- Visit <https://ropsten.etherscan.io/>



01001101  
01010011  
01010000

# Setup - Remix

1. Head to <https://remix.ethereum.org>
2. Click on “Run” in the top left corner
3. Make sure you see your account address and are connected to Ropsten via Injected Web 3



01001101  
01010011  
01010000

Now you're ready to create  
your own Smart Contract!

# Ethereum & Smart Contracts

01001101  
01010011  
01010000

# Internet of Agreements

„Internet is to communication  
what

Ethereum is to agreements“  
(Gavin Wood, Co-Founder Ethereum)

01001101  
01010011  
01010000

# Ethereum Smart Contracts

- Ethereum uses blockchain to implement **arbitrary social contracts** without central server
- Like Bitcoin, accounts with **balances**
- Unlike Bitcoin, accounts can be **contracts**, which hold:
  - **Code** to execute
  - **Storage** for structured data

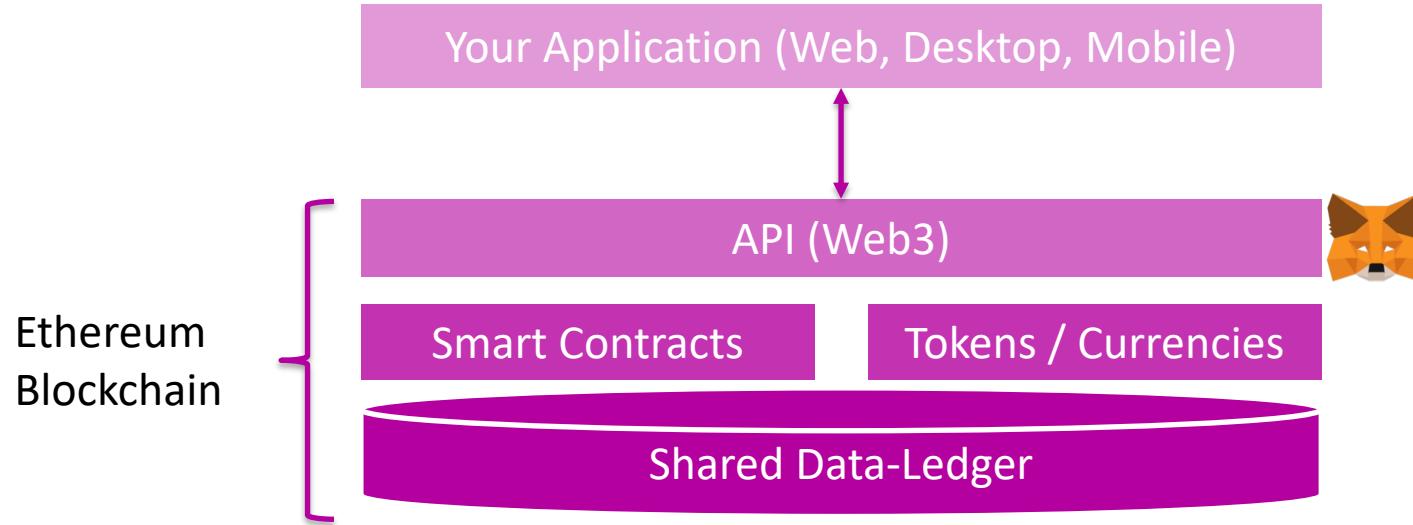
01001101  
01010011  
01010000

# Reasons to use blockchain Smart Contracts

- Trust
- Autonomous
- Redundancy
- Savings
- Speed
- Transparency
- Precision

01001101  
01010011  
01010000

# Smart Contract: Application Architecture



# Solidity

01001101  
01010011  
01010000

# Solidity (programming language)

- Solidity: A contract oriented, high-level programming language for implementing smart contracts
- Influenced by
  - C++
  - Python
  - Javascript



01001101  
01010011  
01010000

# Solidity: Key features

- Statically-typed
- Inheritance
- Libraries
- Complex user-defined types



01001101  
01010011  
01010000

# Solidity: Basics

- Variables
- Functions
- Function modifiers
- Control Structures
- Message Object

01001101  
01010011  
01010000

# Solidity: Structure of a contract

```
/* ----- Example contract ----- */  
  
pragma solidity ^0.4.24  
contract Example {  
    //State variables  
  
    //User defined types: Enums / Structs  
  
    //Events  
  
    //Custom function modifiers  
  
    //Function definitions  
}
```

01001101  
01010011  
01010000

# Solidity: Variables

- State variables are **permantly** stored in **contract storage**
- Variable types:
  - Boolean: bool
  - Integers: int, uint, uint8, ...
  - Ethereum address: address (with members balance & transfer)
  - Arrays: uint[20] someArray

01001101  
01010011  
01010000

# Solidity: Variables (cont.)

- Variable types:
  - Mappings: `mapping (_keyType => _valueType)`
  - String: `string` (dynamically-sized UTF-8 encoded string)

01001101  
01010011  
01010000

# Solidity: Functions

- Function calls can happen internally and externally and have different levels of visibility
- Take set of input parameters and return set of output parameters

01001101  
01010011  
01010000

# Solidity: Functions (Visibility)

- **Visibility:**
  - Public: Function can be called internally or via message
  - Private: Function can only be called from within contract  
(this does not include derived contracts)

01001101  
01010011  
01010000

# Solidity: Functions (Types, Modifiers)

- Important types of functions:
  - View: Function does not modify state (of contract)
  - Payable: Functions that accept Ether sent with function call

01001101  
01010011  
01010000

# Solidity: Message Object

- Special variable which always exists in the global namespace: **msg**
- Important members:
  - **msg.sender**: address that sent the message / function call (address)
  - **msg.value**: Amount of wei sent with message (uint)

# „Hello – World“ Contract

# Greeter Contract

Try it yourself!

01001101  
01010011  
01010000

# Solidity: Further Functionalities

- Custom function modifiers
- Events
- Enum & Structs
- Control structures (if, else, while, do, for, break, continue)

01001101  
01010011  
01010000

# Ethereum: Units of Ether

Name	Conversion Rate
wei	1 wei
kwei	$10^3$ wei
mwei	$10^6$ wei
gwei	$10^9$ wei
szabo	$10^{12}$ wei
finney	$10^{15}$ wei
ether	$10^{18}$ wei

01001101  
01010011  
01010000

# Contract Deployment: General Process

- Solidity compiler (`solcjs`) generates two types of files:
  - Application Binary Interface (ABI)
  - Bytecode
- Deployment: Bytecode is written to Ethereum Blockchain using a transaction

# ABI & Bytecode

01001101  
01010011  
01010000

# Ethereum Network: Gas (Transaction Fees)

- Every transaction is charged with a certain amount of gas (by transaction sender)
- Gas:
  - limits amount of work for the execution of a transaction
  - pays for the execution
- Gasprice: Price per gas in **ether**
- Total TX Fee = gasprice \* gas

# Voting Contract

# Example Use Cases

01001101  
01010011  
01010000

# Use Case I – District0x

**Problem being solved:** the value of companies often depends on the community, but the users cannot participate

**How it works:** kind of its own little government with districts

smart contracts build an open source framework

provides basic functionalities of an online marketplace

getting rid of intermediates

**Smart contract usage:** users can only interact with district0x Network Tokens to protect the network

vote or pay with smart contracts

01001101  
01010011  
01010000

# Use Case II – Grid+

**Problem being solved:** purchase electricity needs a middleman by now

**How it works:** a little iot device now purchases what you need

it buys and sells electricity on behalf of the user

it provides an accounting layer for the energy ecosystem

pushes market signals to the users to

**Smart contract usage:** the smart contracts are used to exchange of kWh to GIRD tokens to money

01001101  
01010011  
01010000

# Use Case III – CryptoKitties

**Problem being solved :** there is never enough cat content!

**How it works:** collect, trade and breed cute kitties

it uses MetaMask to connect to the blockchain

the user can learn how to use smart contracts in a fun way

**Smart contract usage:** generation 0 kitties are distributed as smart contracts

# Microsoft Student Partners

01001101  
01010011  
01010000

# Microsoft Student Community

You want to keep in touch?

Get to know event dates earlier?

Win wildcards for exclusive MSP-events?

Join the Microsoft Student Community!

<https://www.facebook.com/groups/MicrosoftStudentCommunity/>