

Blockchain architectures

From smart contracts to tokens and DApps

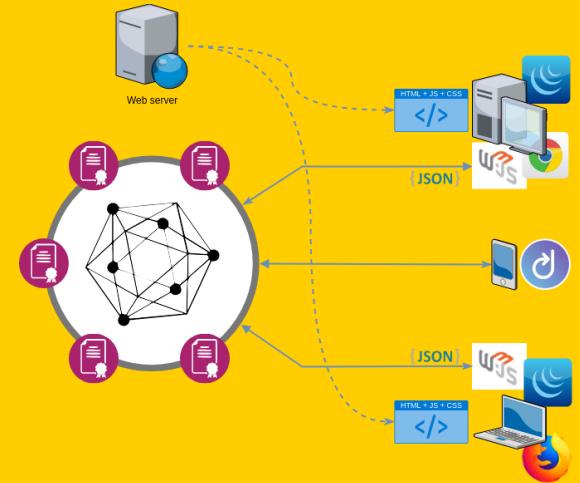


Utrecht
University

Claudio Di Ciccio

c.diciccio@uu.nl

<https://www.uu.nl/staff/CdiCiccio>



Agenda for today



13:15 - 13:20:
Recap

13:20 - 14:00:
Smart Contracts and
DApps

14:15 - 15:00:
Tokens and hands on
Solidity

15:00 - 15:30:
Feedback

15:30 - 17:00:
Architectural debate

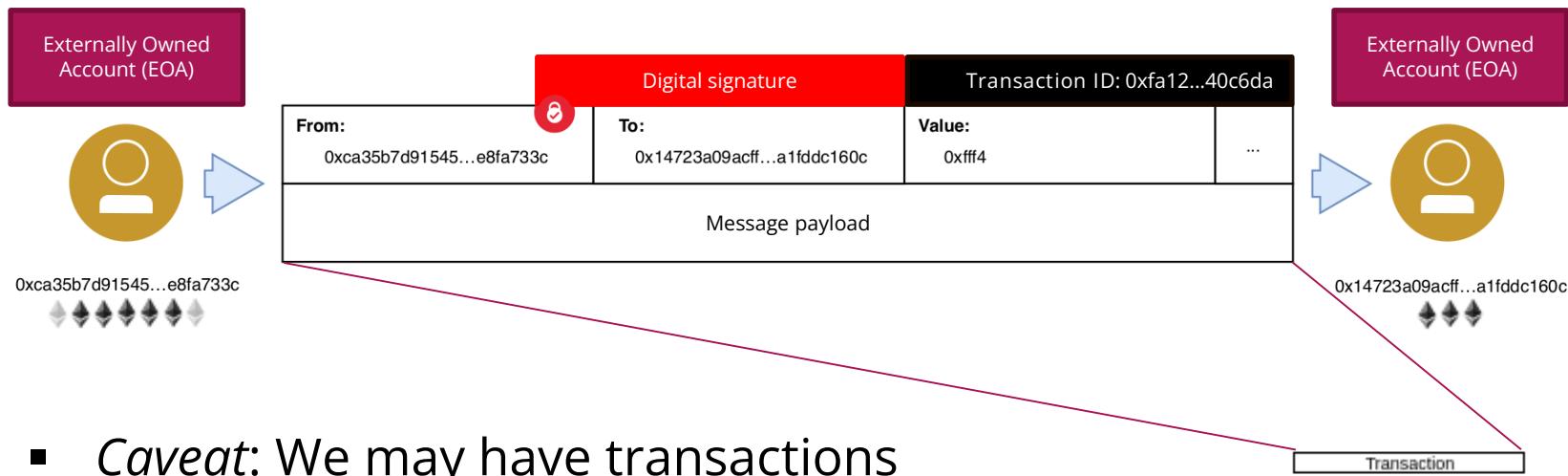


Utrecht
University

Recap and Q&A

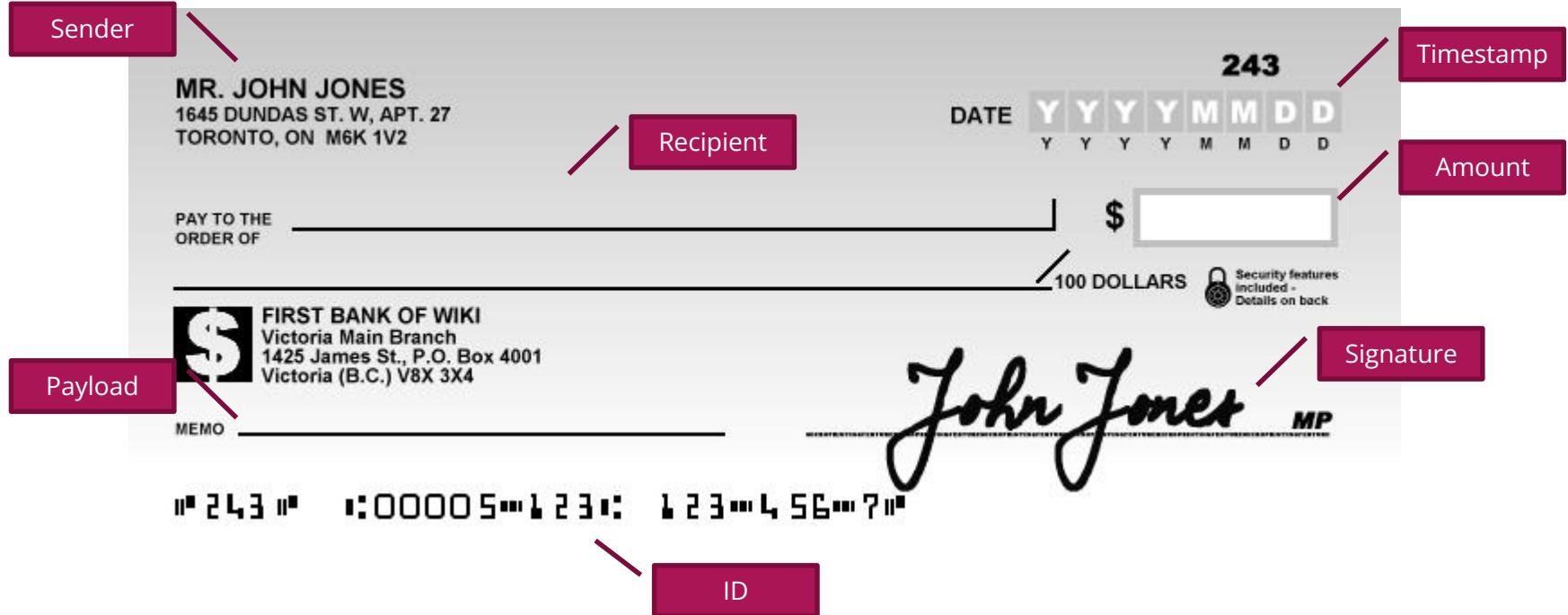
Transaction

- Transfer of (crypto)assets (Ether, Bitcoin, Litecoin, EOS, ...) from account A to account B



- Caveat:* We may have transactions from account A to accounts B, C and A itself or from account A to an unspecified/non-existing/burning-bin accounts

Metaphor: the cheque



Core concepts

- Transactions → Transfer of assets (and code invocation)
- Signatures → Authentication
- Ledger → Transaction ordering
- Distributed architecture → Data persistency
- Hashing → Robustness
- Proof-of-[...] → Publishing rights
- Consensus → Eventual consistency
- Smart contracts → Programmability (and tokens / app coins)

Smart Contracts are codified autonomous agents pieces of code¹

```
1 // SPDX-License-Identifier: CC-BY-SA-4.0
2 pragma solidity >=0.8.0 <0.9.0;
3
4 contract HelloToken {
5     address public minter; // The creator of the contract instance
6     mapping (address => uint) public balances; // The balances in Hello-Tokens
7     uint public constant PRICE = 2000000000; // The price of a Hello Token (2 Gwei)
8
9     constructor() { // Deploys new instances of the smart contract
10         minter = msg.sender; // The sender is the creator
11     }
12
13     function mint() public payable {
14         // Request the minimum amount for a Hello Token, or terminate
15         require(msg.value >= PRICE, "Not enough value for a token!");
16         // Add new Hello Tokens to the balance of the sender
17         balances[msg.sender] += msg.value / PRICE;
18         // The value of the transaction is acquired by the Smart Contract account
19     }
20
21     function transfer(uint amount, address to) public {
22         require(balances[msg.sender] >= amount, "Not enough tokens!");
23         // Decrease the amount from the sender
24         balances[msg.sender] -= amount;
25         // Increase the amount of Hello Tokens to a specified address
26         balances[to] += amount;
27     }
28
29     function terminate() public {
30         // Only the contract creator can terminate this instance
31         require(msg.sender == minter, "You cannot terminate the contract!");
32         // Terminate the contract instance and transfer the balance amount to the creator
33         selfdestruct(payable(minter));
34     }
35 }
```

- Smart Contracts in Ethereum
 - live in the Ethereum environment
 - execute a function when called
 - have direct control over their own balance and key/value storage
 - have their behaviour fully specified by their code

From high-level code to bytecode to bits and bytes

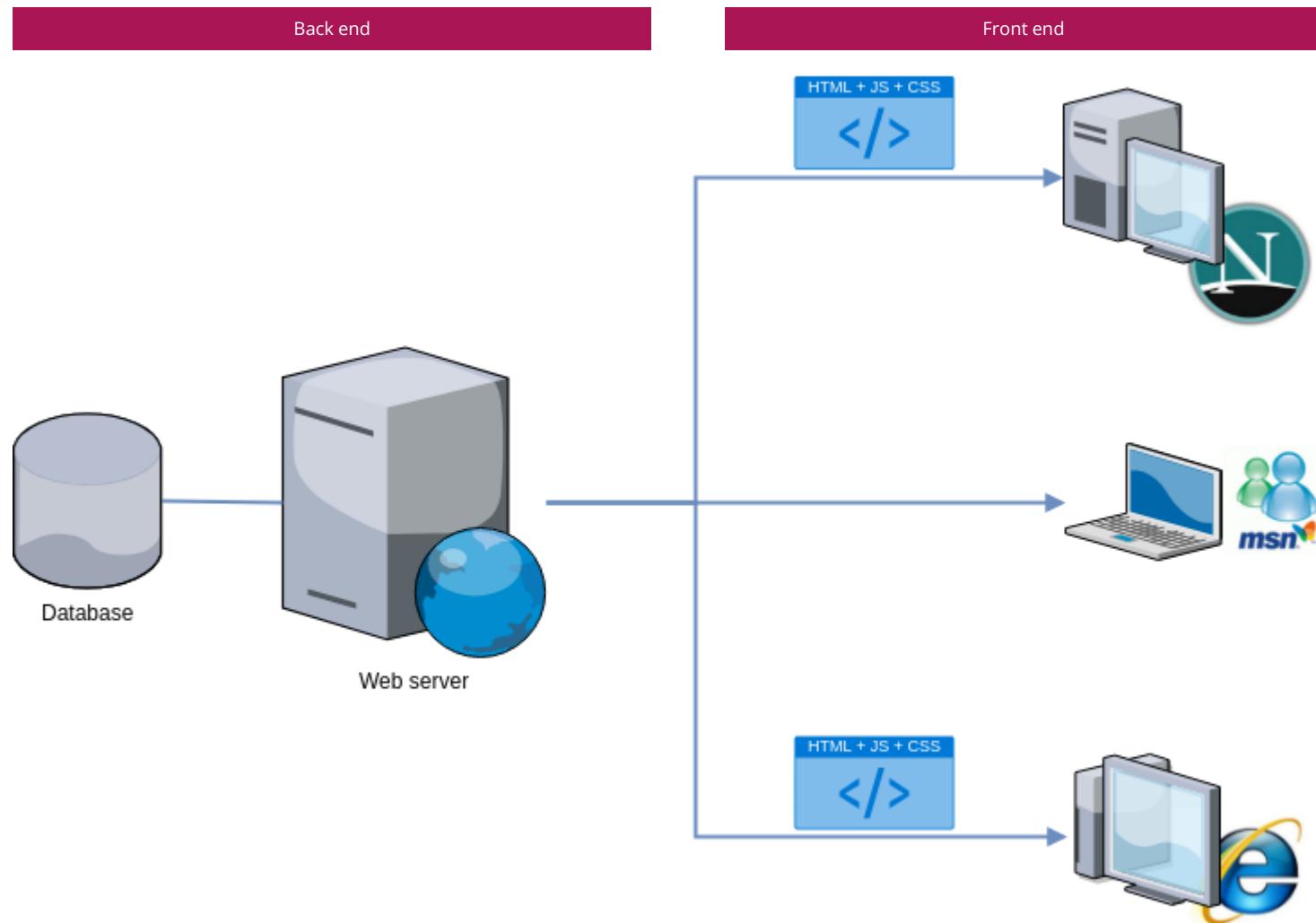
```
1 // SPDX-License-Identifier: CC-BY-SA-4.0
2 pragma solidity >=0.8.0 <0.9.0;
3
4 contract HelloToken {
5     address public minter; // The creator of the contract instance
6     mapping (address => uint) public balances; // The balances in Hello-Tokens
7     uint public constant PRICE = 2000000000; // The price of a Hello Token (2 Gwei)
8
9     constructor() { // Deploys new instances of the smart contract
10         minter = msg.sender; // The sender is the creator
11     }
12
13     function mint() public payable {
14         // Request the minimum amount for a Hello Token, or terminate
15         require(msg.value >= PRICE, "Not enough value for a token!");
16         // Add new Hello Tokens to the balance of the sender
17         balances[msg.sender] += msg.value / PRICE;
18         // The value of the transaction is acquired by the Smart Contract account
19     }
20
21     function transfer(uint amount, address to) public {
22         require(balances[msg.sender] >= amount, "Not enough tokens!");
23         // Decrease the amount from the sender
24         balances[msg.sender] -= amount;
25         // Increase the amount of Hello Tokens to a specified address
26         balances[to] += amount;
27     }
28
29     function terminate() public {
30         // Only the contract creator can terminate this instance
31         require(msg.sender == minter, "You cannot terminate the contract!");
32         // Terminate the contract instance and transfer the balance amount to the creator
33         selfdestruct(payable(minter));
34     }
35 }
```



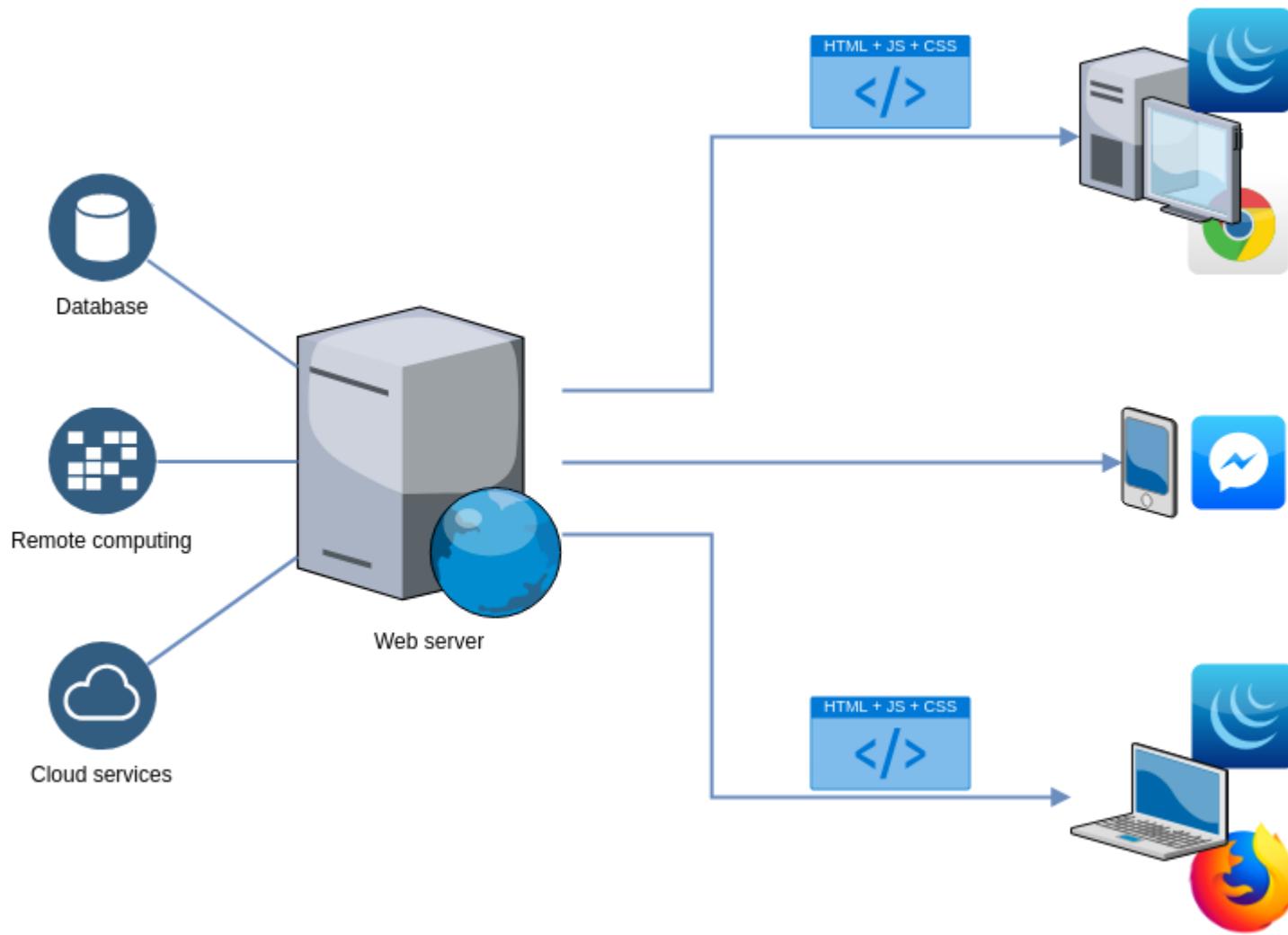
**Utrecht
University**

DApps

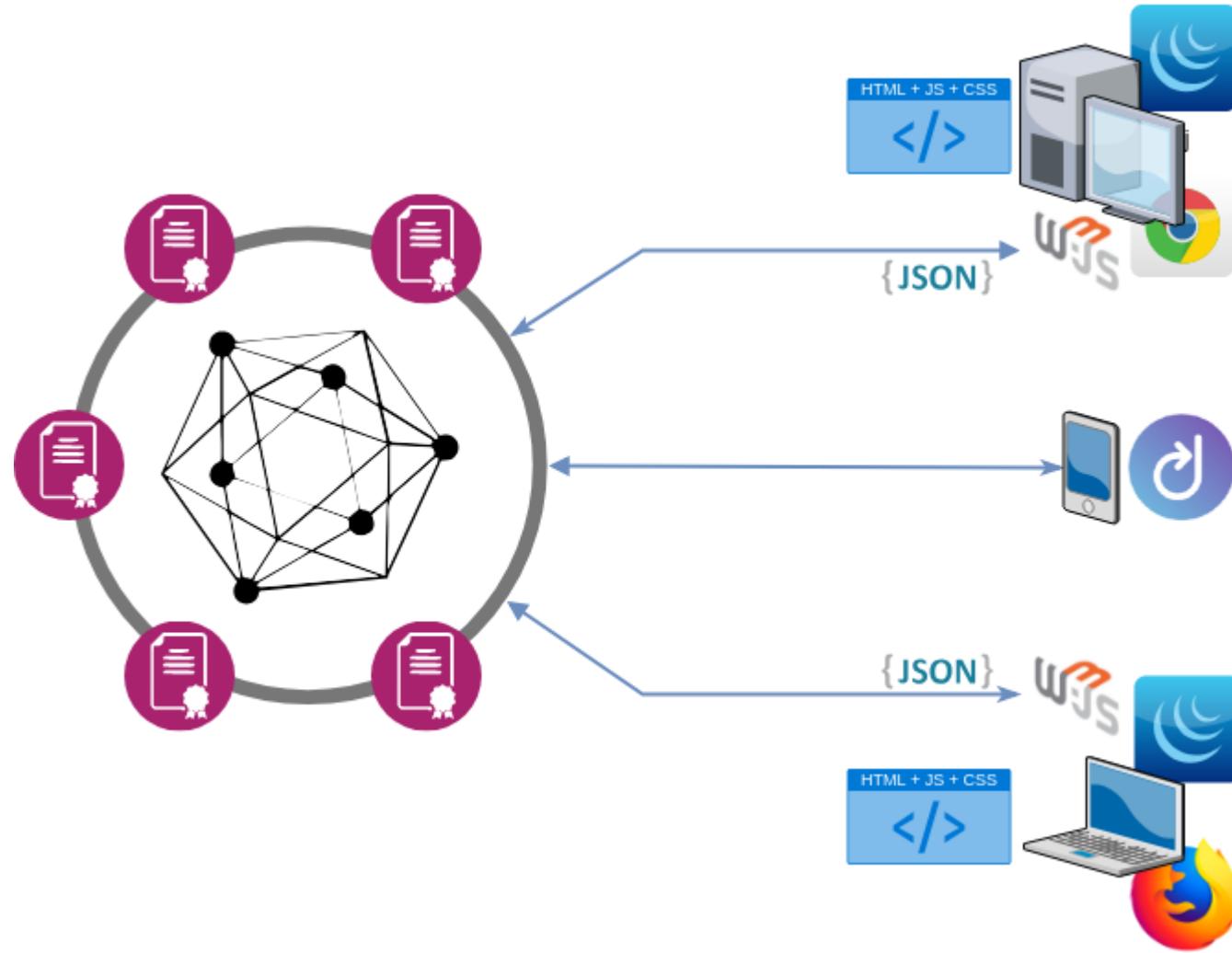
Web 1.0



Web 2.0



Web 3.0



The Ethereum Virtual Machine (EVM)

- Think of the EVM as an emulation of a single, global “computer”
- A globally accessible virtual machine (like a mainframe)
 - in fact, lots of smaller computers
- There is a cost associated to the running of programs on the EVM
 - Please welcome the notion of gas



```
address public minter;
mapping (address => uint) public balances;

// Events allow light clients to react on
// changes efficiently.
event Sent(address from, address to, uint amount);

// This is the constructor whose code is
// run only when the contract is created.
function Coin() {
    minter = msg.sender;
}

function mint(address receiver, uint amount) {
    if (msg.sender != minter) return;
    balances[receiver] += amount;
}

function send(address receiver, uint amount) {
    if (balances[msg.sender] < amount) return;
    balances[msg.sender] -= amount;
    balances[receiver] += amount;
}
```

Table 3-2. This is a complete list of EVM opcodes

0s: Stop and Arithmetic Operations

0x00	STOP	Halts execution.
0x01	ADD	Addition operation.
0x02	MUL	Multiplication operation.
0x03	SUB	Subtraction operation.
0x04	DIV	Integer division operation.
0x05	SDIV	Signed integer.
0x06	MOD	Modulo.
0x07	SMOD	Signed modulo.
0x08	ADDMOD	Modulo.
0x09	MULMOD	Modulo.
0xa	EXP	Exponential operation.
0xb	SIGEXTEND	Extend length of 2s (complement signed integer).

10s: Comparison and Bitwise Logic Operations

0x10	LT	Lesser-than comparison.
0x11	GT	Greater-than comparison.
0x12	SLT	Signed less-than comparison.
0x13	SGT	Signed greater-than comparison.
0x14	EQ	Equality comparison.

(continued)

Smart contracts are pieces of code (not for free)

Name	Value	Description
G_{zero}	0	Nothing paid for operations of the set W_{zero} .
G_{jumpdest}	1	Amount of gas to pay for a JUMPDEST operation.
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} .
$G_{\text{warmaccess}}$	100	Cost of a warm account or storage access.
$G_{\text{accesslistaddress}}$	2400	Cost of warming up an account with the access list.
$G_{\text{accessliststorage}}$	1900	Cost of warming up a storage with the access list.
$G_{\text{coldaccountaccess}}$	2600	Cost of a cold account access.
$G_{\text{coldstorage}}$	2100	Cost of a cold storage access.
G_{asset}	20000	Paid for an SSTORE operation when the storage value is set to non-zero from zero.
G_{reset}	2900	Paid for an SSTORE operation when the storage value's zeroess 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.
$R_{\text{selfdestruct}}$	24000	Refund given (added into refund counter) for self-destructing an account.
$G_{\text{selfdestruct}}$	5000	Amount of gas to pay for a SELFDESTRUCT operation.
G_{create}	32000	Paid for a CREATE operation.
$G_{\text{codedeposit}}$	200	Paid per byte for a CREATE operation to succeed in placing code into state.
G_{calvalue}	9000	Paid for a non-zero value transfer as part of the CALL operation.
$G_{\text{callstipend}}$	2300	A stipend for the called contract subtracted from G_{calvalue} for a non-zero value transfer.
$G_{\text{newaccount}}$	25000	Paid for a CALL or SELFDESTRUCT operation which creates an account.
G_{exp}	10	Partial payment for an EXP operation.
G_{expbyte}	50	Partial payment when multiplied by the number of bytes in the exponent 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.
$G_{\text{txdatazero}}$	4	Paid for every zero byte of data or code for a transaction.
$G_{\text{txdatanonzero}}$	16	Paid for every non-zero byte of data or code for a transaction.
$G_{\text{transaction}}$	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.
$G_{\text{keccak256}}$	30	Paid for each KECCAK256 operation.
$G_{\text{keccak256word}}$	6	Paid for each word (rounded up) for input data to a KECCAK256 operation.
G_{copy}	3	Partial payment for "COPY" operations, multiplied by words copied, rounded up.
$G_{\text{blockhash}}$	20	Payment for each BLOCKHASH operation.

The paradigm

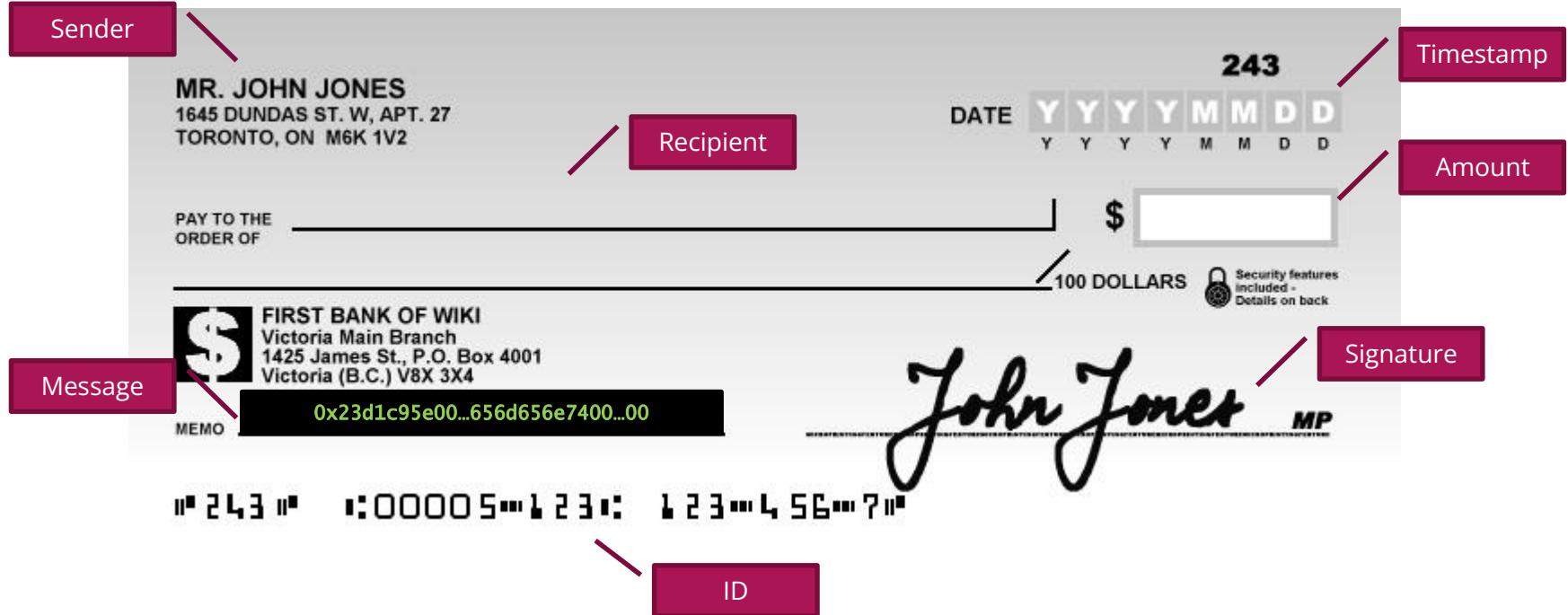
Mainframe



Terminal



Metaphor: the cheque

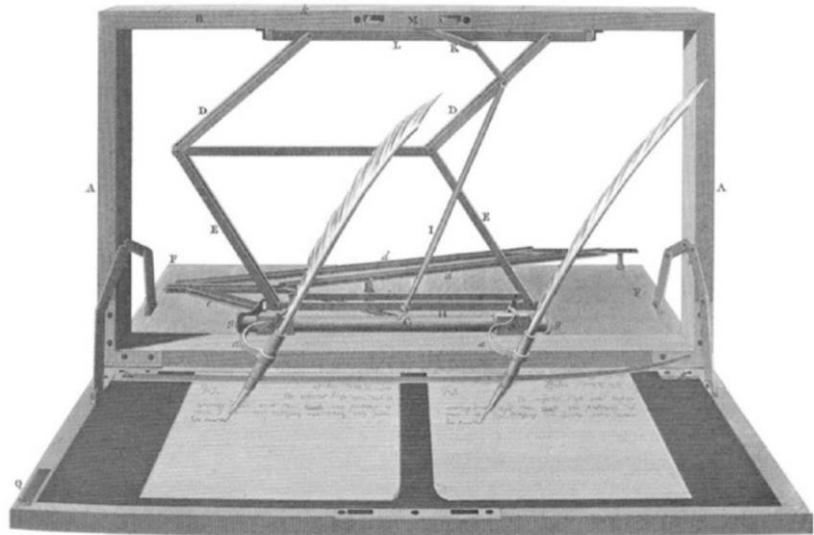


The polygraph machine

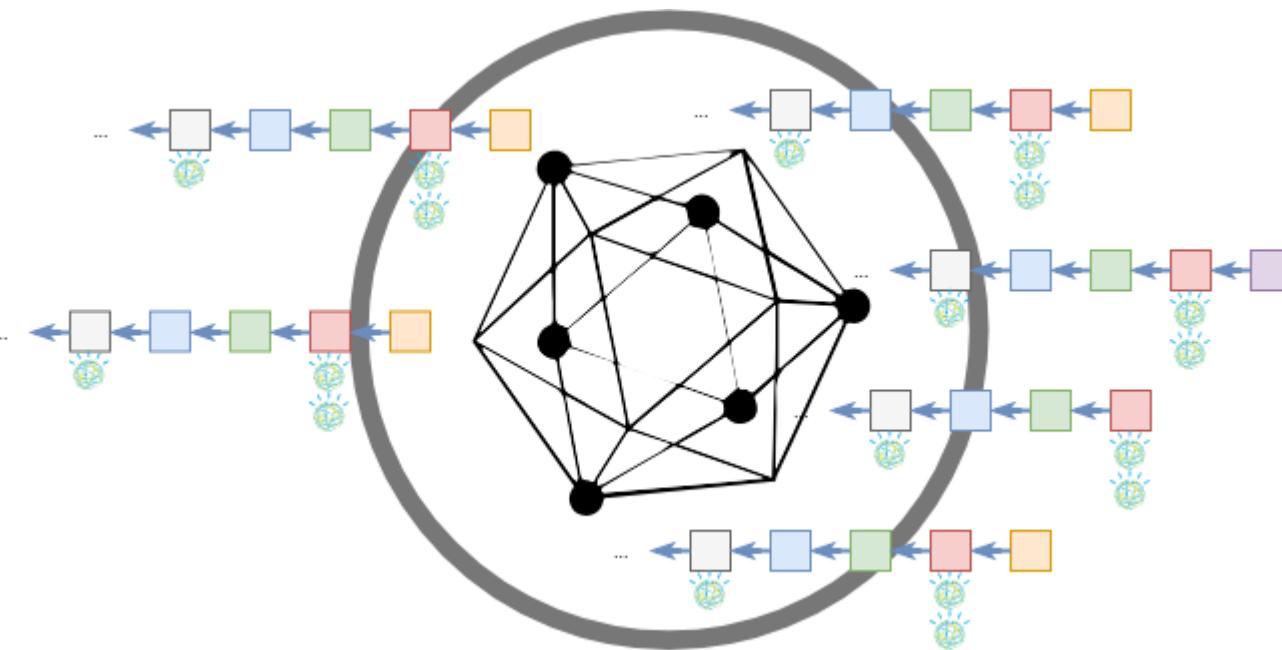
Where are Smart Contracts
executed?

First on the mining nodes.
Then, potentially, on every node!

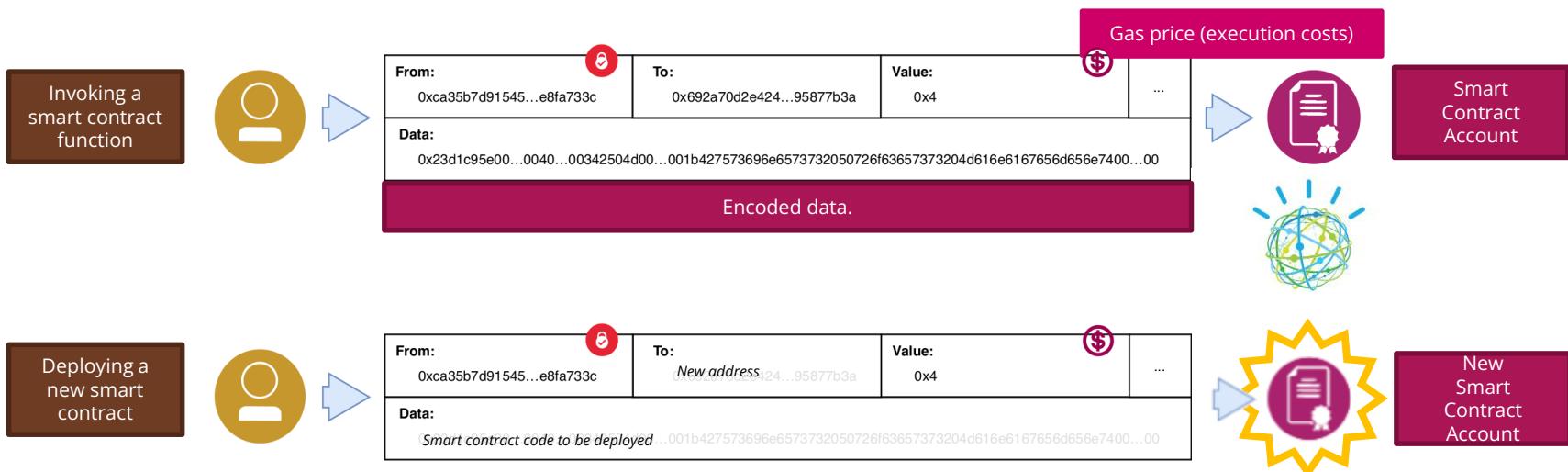
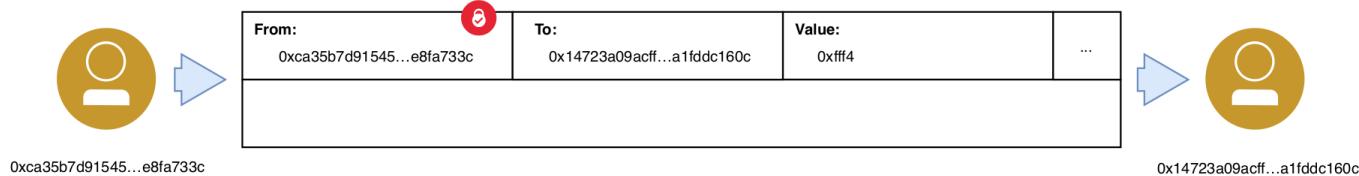
Only absolutely needed
instructions should be in the
code!



Distributed nature

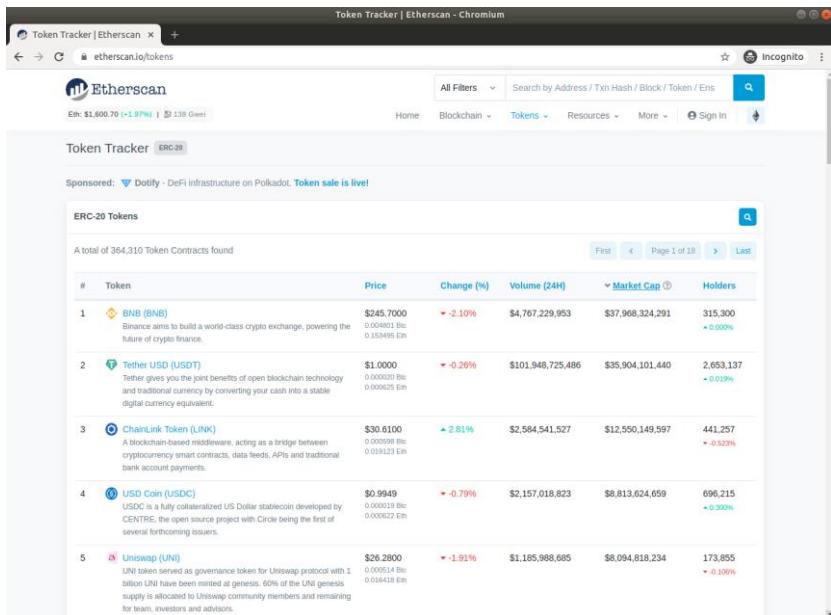


A programmable distributed environment



Smart contracts and tokens

```
1 pragma solidity ^0.4.0;
2
3 contract HelloToken {
4     address public minter;
5     mapping (address => uint) public balance;
6     uint public constant PRICE = 2 finney;
7
8     constructor() public {
9         minter = msg.sender;
10    }
11
12    function mint() public payable {
13        require(msg.value >= PRICE, "Not enough value for a token!");
14        balance[msg.sender] += msg.value / 2 finney;
15    }
16
17    function transfer(uint amount, address to) public {
18        require(balance[msg.sender] >= amount, "Not enough tokens!");
19        balance[msg.sender] -= amount;
20        balance[to] += amount;
21    }
22
23    function terminate() public {
24        require(msg.sender == minter, "You cannot terminate the contract!");
25        selfdestruct(minter);
26    }
27}
```



The screenshot shows a web browser window titled "Token Tracker | Etherscan - Chromium" with the URL "etherscan.io/tokens". The page displays a table of ERC-20 tokens. At the top, it says "A total of 364,310 Token Contracts found". The table has columns: #, Token, Price, Change (%), Volume (24H), Market Cap, and Holders. The data is as follows:

#	Token	Price	Change (%)	Volume (24H)	Market Cap	Holders
1	BNB (BNB) Binance aims to build a world-class crypto exchange, powering the future of crypto finance.	\$245.7000 0.054802 Btc 0.153495 Eth	-2.10%	\$4,767,229,953	\$37,968,324,291	315,300 ▲ 0.00%
2	Tether USD (USDT) Tether gives you the pain benefits of open blockchain technology and traditional currency by converting your cash into a stable digital currency equivalent.	\$1.0000 0.000020 Btc 0.000020 Eth	-0.26%	\$101,948,725,486	\$35,904,101,440	2,053,137 ▲ 0.03%
3	ChainLink Token (LINK) A blockchain-based middleware, acting as a bridge between cryptocurrency smart contracts, data feeds, APIs and traditional bank account payments.	\$30.6100 0.000019 Btc 0.000023 Eth	▲ 2.81%	\$2,584,541,527	\$12,550,149,597	441,257 ▼ -0.52%
4	USD Coin (USDC) USDC is a fully collateralized US Dollar stablecoin developed by CENTRE, the open source project with Circle being the first of several forthcoming issuers.	\$0.9949 0.000019 Btc 0.000022 Eth	-0.79%	\$2,157,018,823	\$8,813,624,659	696,215 ▲ 0.30%
5	Uniswap (UNI) UNI tokens served as governance token for Uniswap protocol with 1 billion UNI have been minted at genesis. 60% of the UNI genesis supply is allocated to Uniswap community members and remaining for team, investors and advisors.	\$26.2800 0.000518 Btc 0.000428 Eth	-1.91%	\$1,185,988,685	\$8,094,818,234	173,855 ▼ -0.36%

Token: A new asset class

!

Notice you may find "app coin" as a synonym for token
and "app token" for cryptocurrency

▪ Multiple functions:

- Currency
 - Large-entity-backed medium of exchange for goods and services
- Commodity
 - Basic good tradeable or exchangeable with goods of the same type
- Utility
 - Satisfaction quantifier for an economic good or service
 - On the Merriam-Wesbter dictionary:
<https://www.merriam-webster.com/dictionary/utility%20token>
- Security
 - Financial instrument that guarantees ownership, credit, or decision power



▪ Different nature:

- Fungible
 - Individual units can be mutually substituted
- Non-fungible
 - Units are unique and not interchangeable
- Semi-fungible
 - Fungible until redeemed or expired

More information on standards:
<https://ethereum.org/en/developers/docs/standards/tokens/>
<https://eips.ethereum.org/EIPS/eip-20>
<https://eips.ethereum.org/EIPS/eip-721>

Reference implementations:
<https://docs.openzeppelin.com/contracts/2.x/api/token/erc20>
<https://docs.openzeppelin.com/contracts/2.x/api/token/erc721>

Some non-fungible tokens (NFTs)

- Crypto collectibles & games
- Property titles
 - Art
 - Collectibles
 - Memorabilia
 - Real Estate
 - Certificates
- Identity
- Keys & passes
- Fractional ownership of physical goods
- Purpose-driven, social-prescribing tokens
 - MCO2 token,
 - TREE token,
 - Plastic Bank token, ...



Your time and passion will become assets.



The Blockchain and the Internet

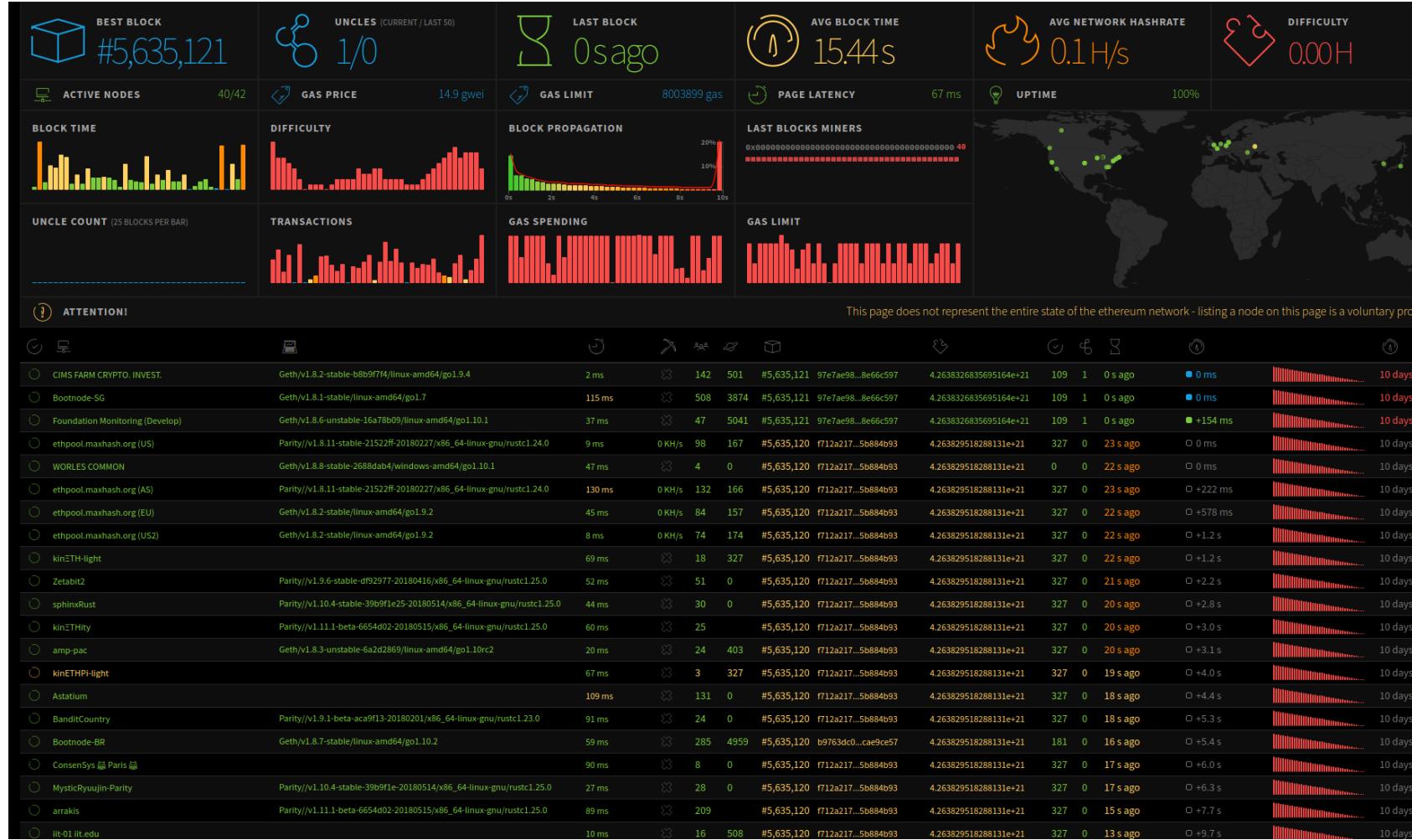


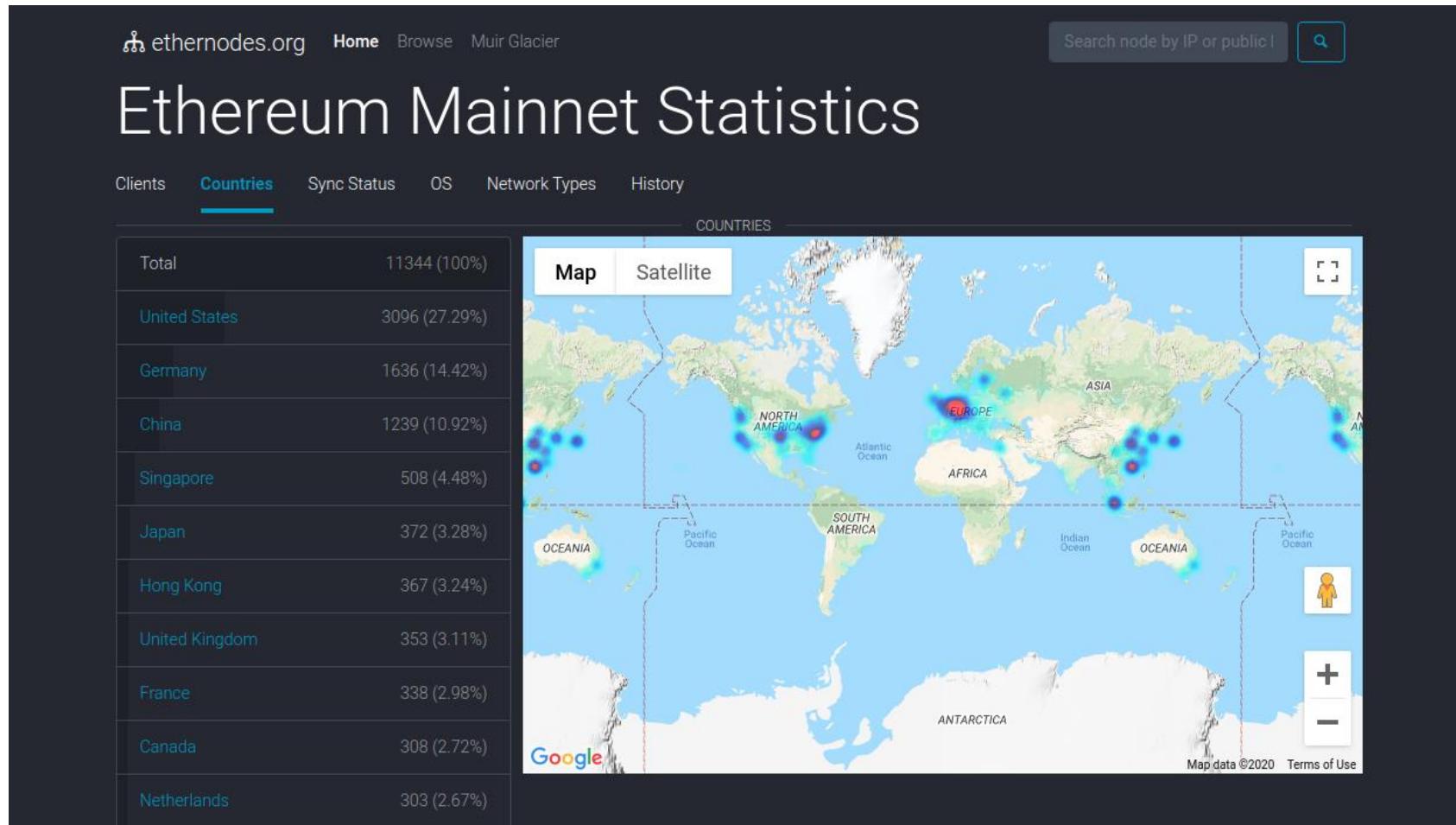
```
student@blockchain: ~/Ethspace/first-dapp-with-truffle
File Edit View Search Terminal Help
student@blockchain:~/Ethspace/first-dapp-with-truffle$ truffle migrate --reset
Using network 'development'.

Running migration: 1_initial_migration.js
Replacing Migrations...
... 0x7d8262b09209822d20a77f63fb64fe04513cc8379fe7822de1047303bf11057e
Migrations: 0xebc4c5940c48331a69cca36409c77cdf4f635ce6
Saving successful migration to network...
... 0x6df36487e47fa0026317ac5479ce8ab0f250eda6ef8f02be45a849217c209ff4
Saving artifacts...
Running migration: 2_deploy_contracts.js
Deploying BitMathGame...
... 0x480259d6fef0f195f2f880784791796495860f218cd46d777ff309a9d67450d7
BitMathGame: 0x3cf4544b0a8fc0aec57414f76e810b7d8bd82622
Saving successful migration to network...
... 0x81d490bb5fc235241338ea889769e394256bd65758b849dbfa9d0cf0f559197e
Saving artifacts...
student@blockchain:~/Ethspace/first-dapp-with-truffle$
```

Decentralised Applications (DApps, or DApps)

- A DApp is a software program
- Its front-end runs on your computer system, but the back-end code runs on a decentralized peer-to-peer network
- In this course, we will only see DApps accessible from web browsers
- CryptoKitties is a DApp
- Dock.io is a DApp
- In the following two slides we have DApps
- Your teamwork's project is a (to-be) DApp







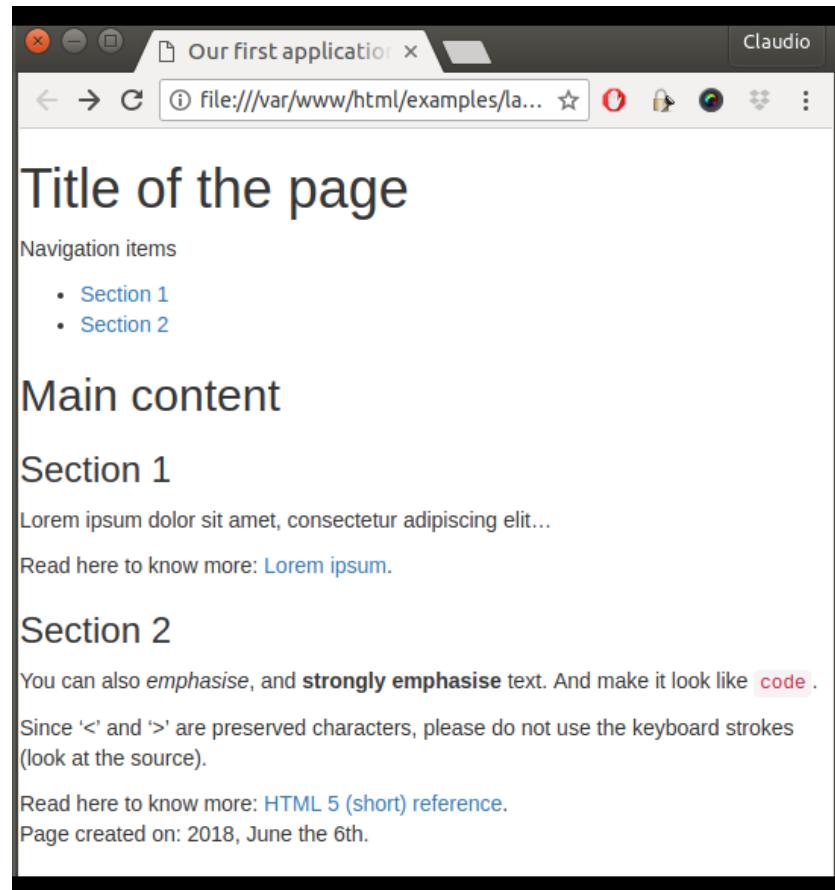
JavaScript

Hypertext Markup Language (HTML)

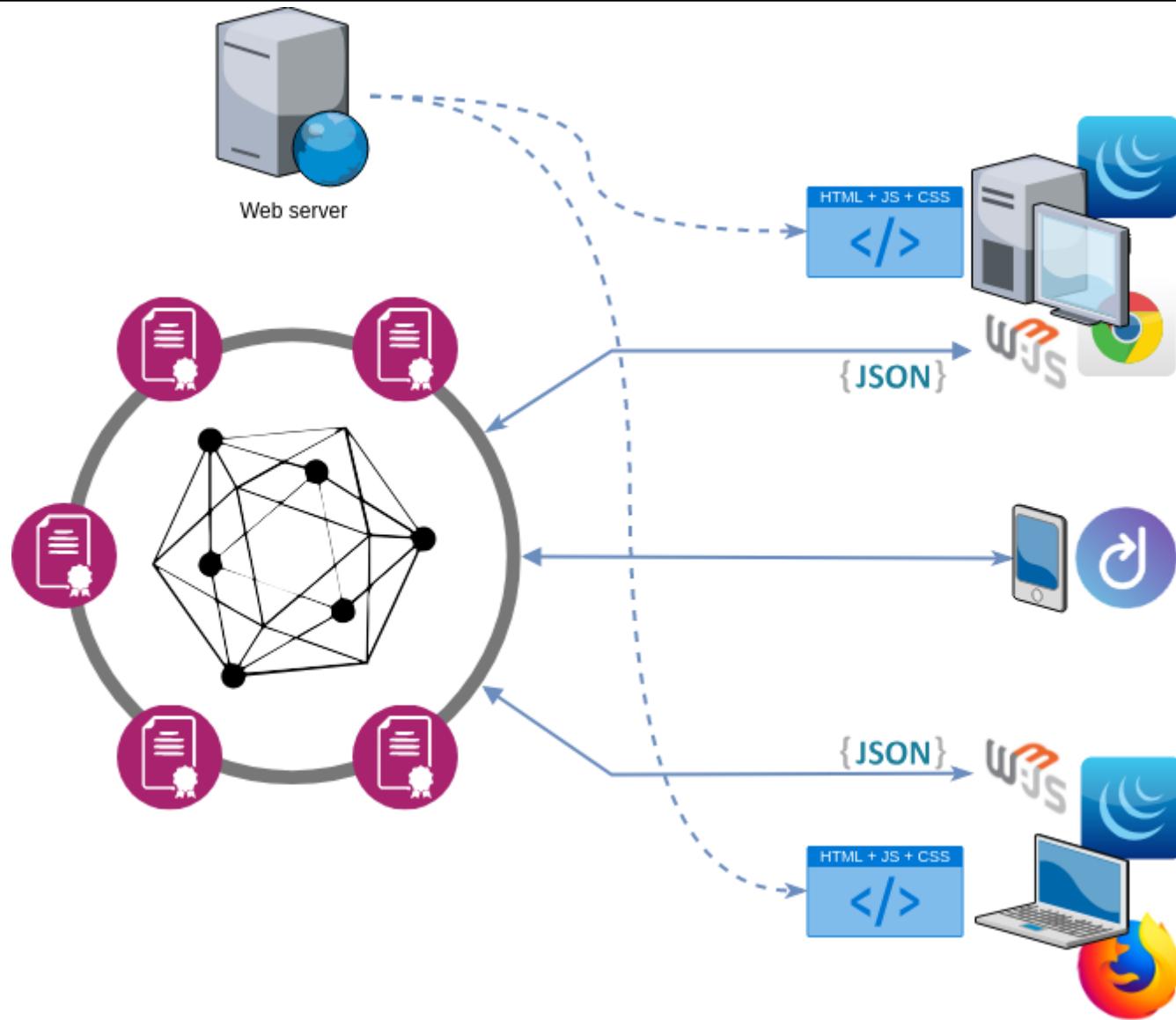
- The standard markup language for creating web pages
 - Based on XML (eXtensible Markup Language)
- It supports, among others:
 - Cascading Style Sheets (CSS)
 - To specify style and layout properties of HTML elements
 - Another W3C standard
 - Javascript (JS)
 - To dynamically change contents of the page
 - Not a standard: ECMA script is
(<https://www.ecma-international.org/publications/standards/Ecma-262.htm>)
 - We will use jQuery to ease our life: <http://api.jquery.com/>
- Suggested links:
 - HTML
 - The W3C standard page for HTML (very detailed):
<https://www.w3.org/TR/html5/>
 - Very nice, interactive tutorial: <https://htmlreference.io/>
 - CSS
 - Mozilla Development Network page for CSS:
https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS

HTML example (file opened locally)

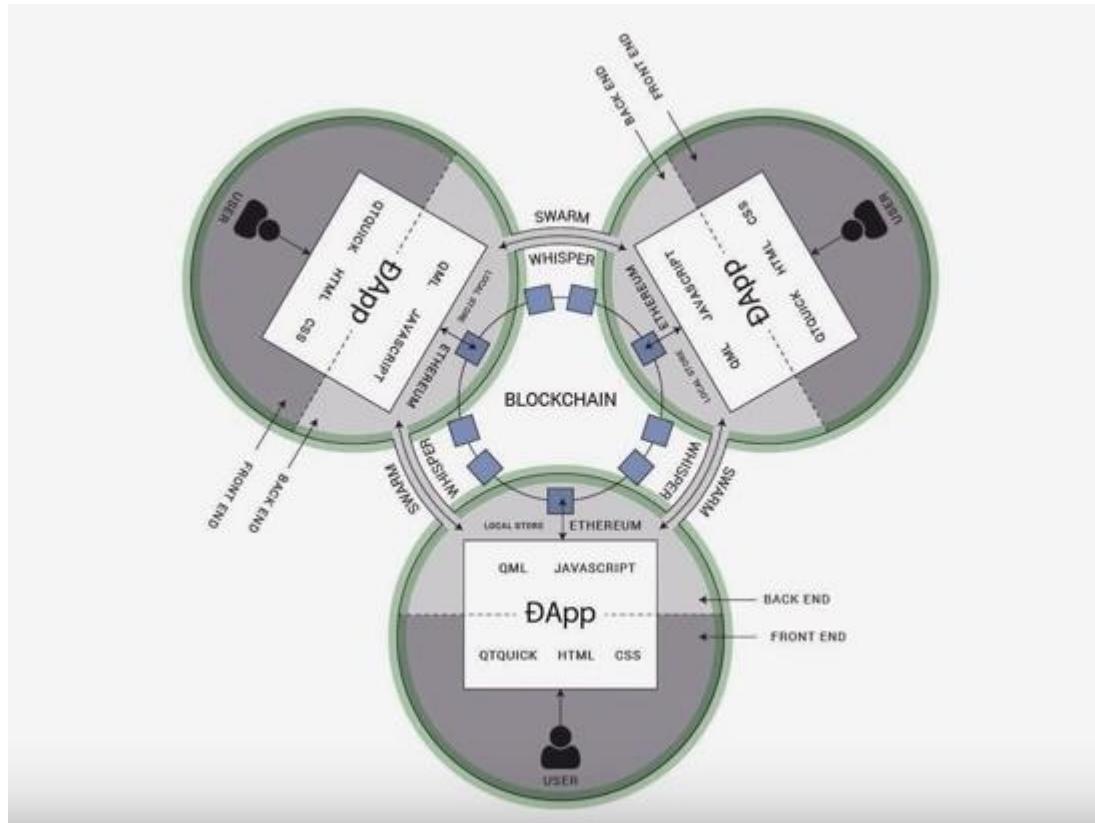
```
1 <!DOCTYPE html>
2 <html lang="en">
3   <meta charset="utf-8">
4   <head>
5     <title>A simple static web document</title>
6     <!-- fancy styling of the page -->
7     <link href="css/bootstrap.min.css" rel="stylesheet">
8   </head>
9   <body>
10    <header>
11      <h1>Title of the page</h1>
12      <nav>
13        <p>Navigation items</p>
14        <ul>
15          <li><a href="#section1">Section 1</a></li>
16          <li><a href="#section2">Section 2</a></li>
17        </ul>
18      </nav>
19    </header>
20    <main>
21      <h2>Main content</h2>
22      <section id="section1">
23        <header><h3>Section 1</h3></header>
24        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit...</p>
25        <footer>Read here to know more:
26          <a href="https://en.wikipedia.org/wiki/Lorem_ipsum">Lorem ipsum</a>.
27        </footer>
28      </section>
29      <section id="section2">
30        <header><h3>Section 2</h3></header>
31        <p>You can also
32          <em>emphasise</em>, and
33          <strong>strongly emphasise</strong> text.
34          And make it look like <code>code</code>.
35        </p>
36        <p>
37          Since '&lt;' and '&gt;' are preserved characters, please
38          do not use the keyboard strokes (look at the source).
39        </p>
40        <footer>Read here to know more:
41          <a href="https://htmlreference.io/">HTML 5 (short) reference</a>.
42        </footer>
43      </section>
44      <section id="section3">
45        <header><h3>Section 3</h3></header>
46        <p>This is the third Section, people!</p>
47        <p>Hah, by the way, I own 1 wei.</p>
48      </section>
49    </main>
50    <footer>
51      Page created on:
52      <time datetime="2020-12-02">2020, December the 2nd</time>.
53    </footer>
54  </body>
55 </html>
```



From where to retrieve HTML+CSS data?



From where to retrieve HTML+CSS data?





Utrecht
University

Some (more) suggested tools

Visual Studio Code

Visual Studio Code Docs Updates Blog API Extensions FAQ Learn [Download](#)

Version 1.74 is now available! Read about the new features and fixes from November.

Code editing. Redefined.

Free. Built on open source. Runs everywhere.

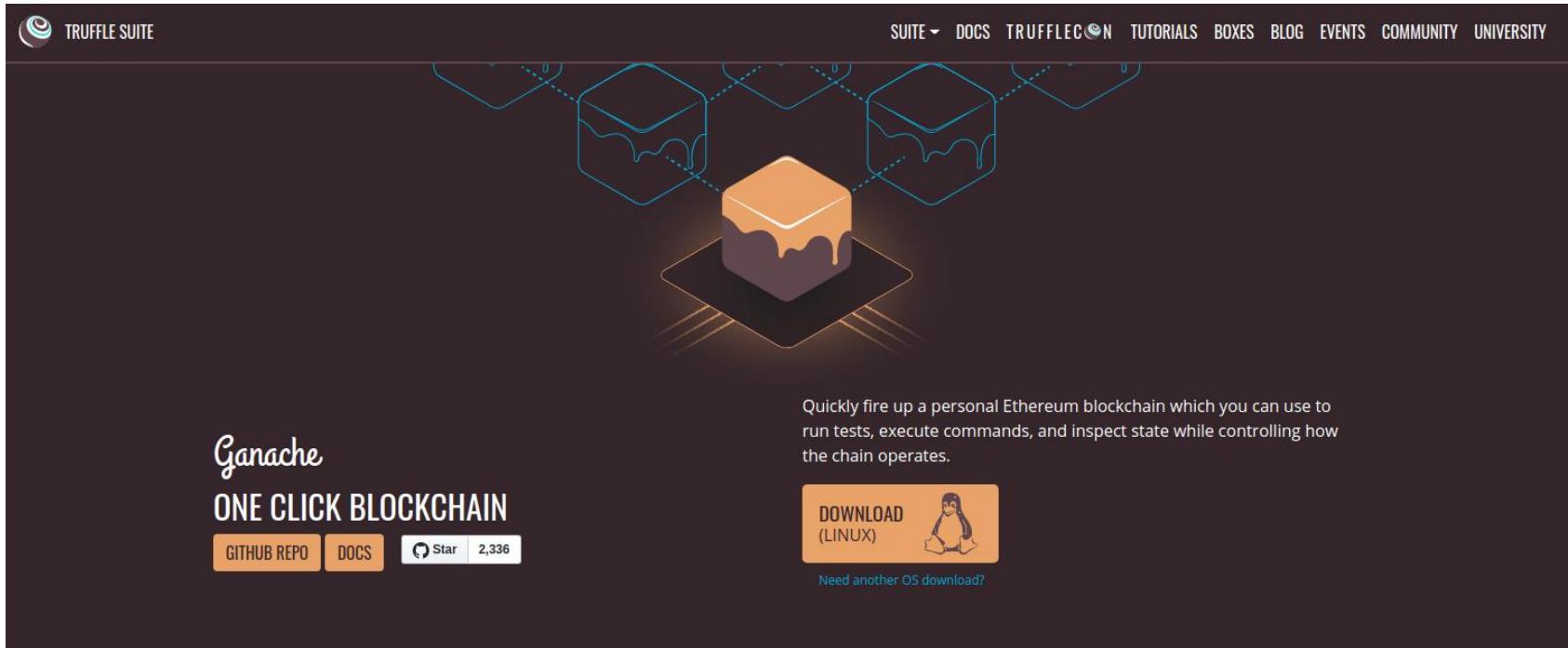
[.deb](#) [.rpm](#) [Web, Insiders edition, or other platforms](#)

By using VS Code, you agree to its [license](#) and [privacy statement](#).

The screenshot shows the Visual Studio Code interface. The code editor has three files: blog-post.js, index.js, and util.js. The blog-post.js file contains code related to Gatsby and GraphQL. The extensions sidebar shows various installed and available extensions like Python, GitLens, C/C++, ESLint, Debugger for Chrome, Language Support, and others. The terminal at the bottom shows a successful build process for a task named 'develop'.



Ganache



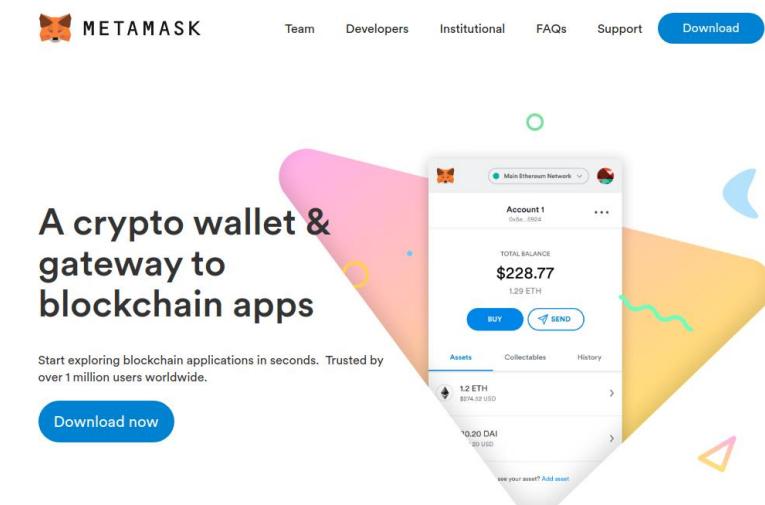
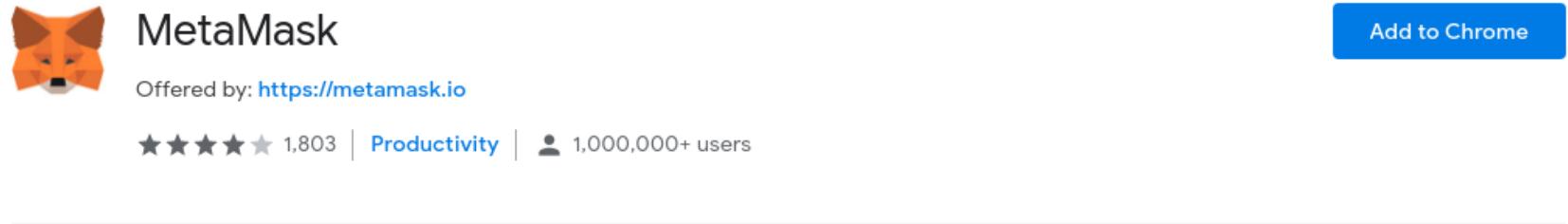
Ganache ships with an internal Javascript implementation of the Ethereum Blockchain which has additional programmatic capabilities

Truffle – <https://www.trufflesuite.com/truffle>

The screenshot shows the official Truffle website. At the top left is the Truffle logo (a stylized green and white swirl). The top right features a navigation bar with links: SUITE ▾, DOCS, TRUFFLECON, TUTORIALS, BOXES, BLOG, EVENTS, and COMMUNITY. Below the header, there's a large dark background image featuring a 3D-style Truffle logo with glowing green lines. To the left of the logo is a semi-transparent dark box containing a snippet of Solidity smart contract code. The code includes annotations like 'public', 'msg.sender', 'onlyOwner', 'require', and 'transferOwnership'. To the right of the logo, the word 'TRUFFLE' is written in large, bold, white capital letters, followed by 'SMART CONTRACTS MADE SWEETER' in a smaller white font. Below this text is a brief description: 'Truffle is the most popular development framework for Ethereum with a mission to make your life a whole lot easier.' A teal button at the bottom left contains the command 'npm install truffle -g'. At the bottom right, there are links for 'GITHUB REPO' (in a teal box), 'DOCS' (in a white box), and a 'Star' button with the number '9,221'.

MetaMask – <https://metamask.io/>

Home > Extensions > MetaMask



- Allows Ethereum DApps to be run in the browser without the need to run a full Ethereum node.
- Includes a (secure) identity vault, providing a user interface to manage identities on different sites and sign blockchain transactions



**Utrecht
University**

Tokens

Your brand new token in 5 minutes or less

The screenshot illustrates the process of creating a simple Ethereum token (Hello Token) using Solidity, MetaMask, and a web-based front-end.

Solidity Code: The code defines a contract named `HelloToken` with a minter account, a mapping of addresses to balances, and three functions: `mint`, `transfer`, and `terminate`.

```
pragma solidity ^0.4.0;
contract HelloToken {
    address public minter;
    mapping (address => uint) public balance;
    uint public constant PRICE = 2 finney;
    constructor() public {
        minter = msg.sender;
    }
    function mint() public payable {
        require(msg.value >= PRICE);
        balance[msg.sender] += msg.value;
    }
    function transfer(uint amount, address to) public {
        require(balance[msg.sender] >= amount);
        balance[msg.sender] -= amount;
        balance[to] += amount;
    }
    function terminate() public {
        require(msg.sender == minter);
        selfdestruct(minter);
    }
}
```

MetaMask Notification: A modal window from MetaMask displays a transaction confirmation for a `mint` operation. It shows the amount (0.005), the gas fee (0.004021), and the total cost (0.009021). The user has the option to `CONFIRM` or `REJECT` the transaction.

Web Interface: The browser window shows the `Hello Token!` application. It features a title "The Hello Token", a heading "Buy Hello Tokens!", and a sub-section "Minting form". A text input field asks "How many tokens do you want?" with a value of 5, and a note below it says "of 2 finneys!". Below this is another section titled "Transfer tokens" with a "Transfer form" and a "Transfer!" button. The "Status" section at the bottom displays the account address (0xd1d993d57ec011b8dbff0dace6705e91a24423df), the current balance (7), and the minter's address (0x13ee11549abb691dc8d1a9c2c91d4d18e5585ea5).

Tokens tokens tokens tokens



No comment

The screenshot shows the Twitter homepage with a search bar at the top. Below it, a "Trends for you" section highlights "Harry Potter" as a trending topic. The main content area displays "What's happening" in various leagues:

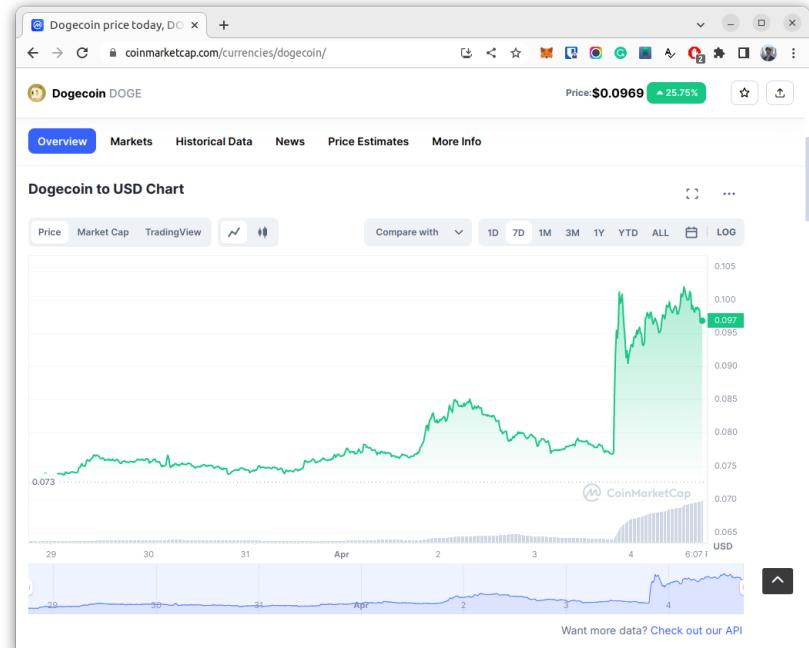
- Premier League: Everton 1 vs Tottenham Hotspur 1
- MLB: Cleveland Guardians 12 vs Oakland Athletics 11
- MLB: Arizona Diamondbacks 4

A blue banner at the bottom encourages users to "Don't miss what's happening" and "People on Twitter are the first to know." There are "Log In" and "Sign up" buttons.

New to Twitter?
Sign up now to get your own personalized timeline!

[Sign up with Google](#)
[Sign up with Apple](#)
[Create account](#)

By signing up, you agree to the [Terms of Service](#) and [Privacy Policy](#), including [Cookie Use](#).



Some non-fungible tokens (NFTs)

- Crypto collectibles & games
- Property titles
 - Art
 - Collectibles
 - Memorabilia
 - Real Estate
 - Certificates
- Identity
- Keys & passes
- Fractional ownership of physical goods
- Purpose-driven, social-prescribing tokens
 - MCO2 token,
 - TREE token,
 - Plastic Bank token, ...



Your time and passion will become assets.



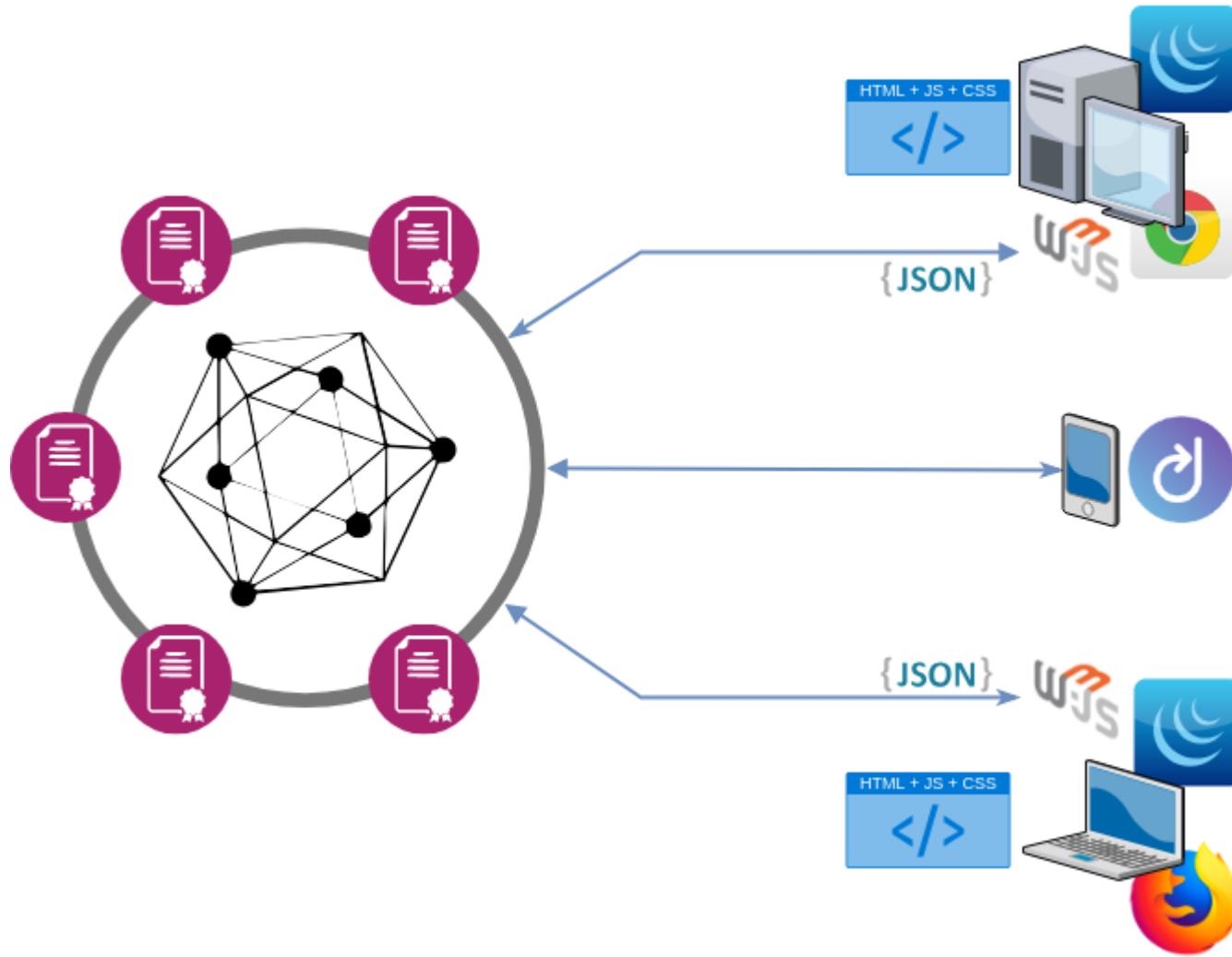
Tokens are not cryptofuel: the Steemit example

▪ A blockchain-based blogging and social media website

- Steem Token (STEEM): Access token
 - Gateway token
 - Instantly transferable
 - Does not incentivize behaviour directly
- Steem Power (SP): Reputation token
 - 1:1 convert ratio with Steem token
 - Needed to make posts, upvotes
 - One can earn SP with upvotes, posts, comments, delegation
 - Non transferrable
- Steem Dollars (SBD): 1 Steem Dollar = 1 USD
 - Stable coin, to act as store of value
 - Earn interest p.a. (only if market price is not higher than US Dollar)
 - Transferrable
 - Pay bots for liking contents



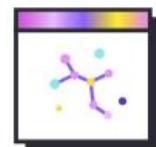
Decentralised application (DApp)





Some interesting effects of Turing completeness

Things may go out of control



CryptoKitties

Collect and breed furrever friends!



Get your own Kitty

Buy & sell cats with our community

Create collections & earn rewards

Breed adorable cats & unlock rare traits

Crack puzzles alongside other players

Chase limited edition Fancy cats

Play games in the KittyVerse

"CryptoKitties is a game centered around breedable, collectible, and oh-so-adorable creatures we call CryptoKitties! Each cat is one-of-a-kind and 100% owned by you; it cannot be replicated, taken away, or destroyed."

Still code, under the hood

The screenshot shows a browser window titled "Ethereum Accounts, Address And Contracts - Chromium". The URL is <https://etherscan.io/address/0x06012c8cf97bead5deae237070f9587f8e7a266d#code>. The page displays a green checkmark indicating "Contract Source Code Verified (Exact Match)". It shows the following details:

- Contract Name: KittyCore
- Compiler Text: v0.4.18+commit.9cf6e910
- Optimization Enabled: Yes
- Runs (Optimiser): 200

The "Contract Source Code" section contains the Solidity code for the `KittyCore` contract, with line numbers 46 through 68. To the right, the corresponding assembly code is shown, starting from line 234. The assembly code is highly optimized, reflecting the byte-packing rules used by Ethereum.

```
// @author Dieter Shirley <det@axionzen.co> (https://github.com/dete)
contract ERC721 {
    // Required methods
    function totalSupply() public view returns (uint256 total);
    function balanceOf(address _owner) public view returns (uint256 balance);
    function ownerOf(uint256 _tokenId) external view returns (address owner);
    function approve(address _to, uint256 _tokenId) external;
    function transfer(address _to, uint256 _tokenId) external;
    function transferFrom(address _from, address _to, uint256 _tokenId) external;

    // Events
    event Transfer(address from, address to, uint256 tokenId);
    event Approval(address owner, address approved, uint256 tokenId);

    // Optional
    // function name() public view returns (string name);
    // function symbol() public view returns (string symbol);
    // function tokensOfOwner(address _owner) external view returns (uint256[] tokenIds);
    // function tokenMetadata(uint256 _tokenId, string _preferredTransport) public view returns (string infoUrl);

    // ERC-165 Compatibility (https://github.com/ethereum/EIPs/issues/165)
    function supportsInterface(bytes4 _interfaceID) external view returns (bool);
}
```

```
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280

    /// @dev The main Kitty struct. Every cat in CryptoKitties is represented by a copy
    /// of this structure, so great care was taken to ensure that it fits neatly into
    /// exactly two 256-bit words. Note that the order of the members in this structure
    /// is important because of the byte-packing rules used by Ethereum.
    /// Ref: http://solidity.readthedocs.io/en/develop/miscellaneous.html
struct Kitty {
    /// The Kitty's genetic code is packed into these 256-bits, the format is
    /// sooper-sekret! A cat's genes never change.
    uint256 genes;

    /// The timestamp from the block when this cat came into existence.
    uint64 birthTime;

    /// The minimum timestamp after which this cat can engage in breeding
    /// activities again. This same timestamp is used for the pregnancy
    /// timer (for matrons) as well as the siring cooldown.
    uint64 cooldownEndBlock;

    /// The ID of the parents of this kitty, set to 0 for gen0 cats.
    /// Note that using 32-bit unsigned integers limits us to a "mere"
    /// 4 billion cats. This number might seem small until you realize
    /// that Ethereum currently has a limit of about 500 million
    /// transactions per year! So, this definitely won't be a problem
    /// for several years (even as Ethereum learns to scale).
    uint32 matronId;
    uint32 sireId;

    /// Set to the ID of the sire cat for matrons that are pregnant,
    /// zero otherwise. A non-zero value here is how we know a cat
    /// is pregnant. Used to retrieve the genetic material for the new
    /// kitten when the birth transpires.
    uint32 siringWithId;

    /// Set to the index in the cooldown array (see below) that represents
    /// the current cooldown duration for this Kitty. This starts at zero
    /// for gen0 cats, and is initialized to floor(generation/2) for others.
    /// Incremented by one for each successful breeding action, regardless
    /// of whether this cat is acting as matron or sire.
    uint16 cooldownIndex;

    /// The "generation number" of this cat. Cats minted by the CK contract
    /// for sale are called "gen0" and have a generation number of 0. The
    /// generation number of all other cats is the larger of the two generation
    /// numbers of their parents, plus one.
    /// (i.e. max(matron.generation, sire.generation) + 1)
    uint16 generation;
}
```

Cryptokitties use (secret) genetic algorithms for breeding simulation

EthFiddle Security Audit Share Login

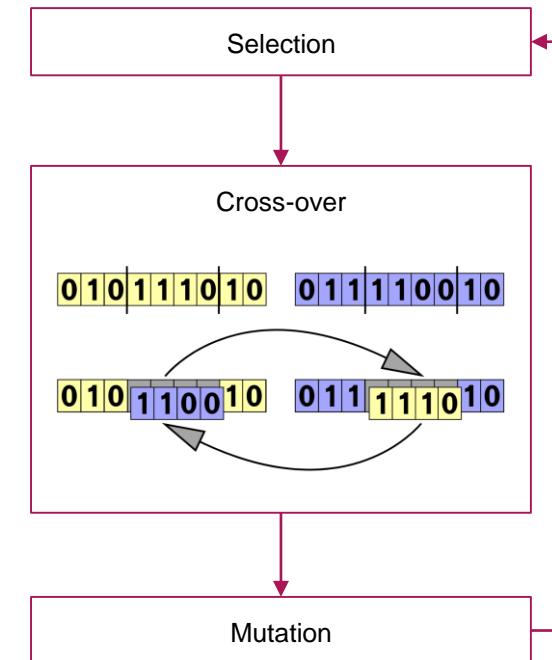
```
1022 function giveBirth(uint256 _matronId)
1023     external
1024     whenNotPaused
1025     returns(uint256)
1026 {
1027     // Grab a reference to the matron in storage.
1028     Kitty storage matron = kitties[_matronId];
1029
1030     // Check that the matron is a valid cat.
1031     require(matron.birthTime != 0);
1032
1033     // Check that the matron is pregnant, and that its time has come!
1034     require(_isReadyToGiveBirth(matron));
1035
1036     // Grab a reference to the sire in storage.
1037     uint256 sireId = matron.siringWithId;
1038     Kitty storage sire = kitties[sireId];
1039
1040     // Determine the higher generation number of the two parents
1041     uint16 parentGen = matron.generation;
1042     if (sire.generation > matron.generation) {
1043         parentGen = sire.generation;
1044     }
1045
1046     // Call the sooper-sekret gene mixing operation.
1047     uint256 childGenes = geneScience.mixGenes(matron.genes, sire.genes, matron cooldownEndBlock - 1);
1048
1049     // Make the new kitten!
1050     address owner = kittyIndexToOwner[_matronId];
1051     uint256 kittenId = _createKitty(_matronId, matron.siringWithId, parentGen + 1, childGenes, owner);
```

Cryptokitties use (secret) genetic algorithms for breeding simulation

EthFiddle

Security Audit Share Login

```
1022 function giveBirth(uint256 _matronId)
1023     external
1024     whenNotPaused
1025     returns(uint256)
1026 {
1027     // Grab a reference to the matron in storage.
1028     Kitty storage matron = kitties[_matronId];
1029
1030     // Check that the matron is a valid cat.
1031     require(matron.birthTime != 0);
1032
1033     // Check that the matron is pregnant, and that its time has come!
1034     require(_isReadyToGiveBirth(matron));
1035
1036     // Grab a reference to the sire in storage.
1037     uint256 sireId = matron.siringWithId;
1038     Kitty storage sire = kitties[sireId];
1039
1040     // Determine the higher generation number of the two parents
1041     uint16 parentGen = matron.generation;
1042     if (sire.generation > matron.generation) {
1043         parentGen = sire.generation;
1044     }
1045
1046     // Call the sooper-sekret gene mixing operation.
1047     uint256 childGenes = geneScience.mixGenes(matron.genes, sire.genes, matron cooldownEndBlock - 1);
1048
1049     // Make the new kitten!
1050     address owner = kittyIndexToOwner[_matronId];
1051     uint256 kittenId = _createKitty(_matronId, matron.siringWithId, parentGen + 1, childGenes, owner);
```



Cryptokitties use (secret) genetic algorithms

The screenshot shows the EthFiddle interface with the following details:

- Left Panel (Solidity Code):** Displays the Solidity code for the `giveBirth` function. The code handles the birth process, including checking the matron's validity, ensuring it's pregnant, determining the higher generation number of parents, mixing genes, and creating a new kitten.
- Right Panel (Assembly View):** Shows the assembly code generated for the `giveBirth` function. The assembly is highly optimized, using various Ethereum-specific instructions like `CALLCODE`, `CREATE2`, and `SHL`.

Cryptokitties are greedy for gas

The screenshot shows the ETH Gas Station website at <https://ethgasstation.info/gasguzzlers.php>. The main content area displays a table titled "Top 100 ETH Contracts By Gas Used Over Last 1,500 Blocks". The table has columns for Address, ID, and Pct Total Gas Used. The data shows that the top contract, with address 0xae9b8e05c22bae74d1e8db82c4af122b18050bd4, used 3.28% of the total gas. Other prominent entries include TxPool Vision, Low Gas Price Watch List, Gas Burners, FAQ, and External Links.

Address	ID	Pct Total Gas Used
0xae9b8e05c22bae74d1e8db82c4af122b18050bd4		3.28
0x667061051d70b8b4077adbb107d1d98d05a33a4a		2.87
0x3401cab9bee49bc76e13a8a09619e53d45c0af0		2.86
0xba5f11b16b155792cf3b2e6880e8706859a8aeb6		2.04
0x0d152b9ee87ebae179f64c067a966dd716c50742		1.82
0x2a0c0dbecc7e4d658f48e01e3fa353f44050c208	IDEX	1.54
0xd1ceeeee83f8bcf3bedad437202b6154e9f5405		1.46
0x105631c6cddb84d12fa916f0045b1f97ec9c268		1.33
0x828bf05d86c6fe30cd6de9f932890fd7aec987e		1.15
0xa52e014b3f5cc48287c2d483a3e026c32cc76e6d		1.03
0x06012c8cf97bead5deae237070f9587f8e7a266d	Cryptokitties	0.89



**Utrecht
University**

Hello Token

HelloToken (cyptoeconomic version of Hello World)

```
1 // SPDX-License-Identifier: CC-BY-SA-4.0
2 pragma solidity >=0.4.0 <0.9.0;
3
4 contract HelloToken {
5     address public minter;
6     mapping (address => uint) public balance;
7     uint public constant PRICE = 2 * 1e15;
8     // uint public constant PRICE = 2 finney;
9     // finney is no longer a supported denomination since Solidity v.0.7.0
10
11 constructor() {
12     minter = msg.sender;
13 }
14
15 function mint() public payable {
16     require(msg.value >= PRICE, "Not enough value for a token!");
17     balance[msg.sender] += msg.value / PRICE;
18     // Guess guess, where does the remainder of the msg.value end?
19 }
20
21 function transfer(uint amount, address to) public {
22     require(balance[msg.sender] >= amount, "Not enough tokens!");
23     balance[msg.sender] -= amount;
24     balance[to] += amount;
25 }
26
27 function terminate() public {
28     require(msg.sender == minter, "You cannot terminate the contract!");
29     selfdestruct(payable(minter));
30 }
31 }
```

Solidity, bytecode and bytes

```

1 // SPDX-License-Identifier: CC-BY-SA-4.0
2 pragma solidity >=0.4.0 <0.9.0;
3
4 contract HelloToken {
5     address public minter;
6     mapping (address => uint) public balance;
7     uint public constant PRICE = 2 * 1e15;
8     // uint public constant PRICE = 2 finney;
9     // finney is no longer a supported denomination since Solidity v.0.7.0
10
11 constructor() {
12     minter = msg.sender;
13 }
14
15 function mint() public payable {
16     require(msg.value >= PRICE, "Not enough value for a token!");
17     balance[msg.sender] += msg.value / PRICE;
18     // Guess guess, where does the remainder of the msg.value end?
19 }
20
21 function transfer(uint amount, address to) public {
22     require(balance[msg.sender] >= amount, "Not enough tokens!");
23     balance[msg.sender] -= amount;
24     balance[to] += amount;
25 }
26
27 function terminate() public {
28     require(msg.sender == minter, "You cannot terminate the contract!");
29     selfdestruct(payable(minter));
30 }
31 }
```

The diagram illustrates the relationship between Solidity code, assembly, and bytecode. The left side shows the Solidity source code for a simple token contract. The middle section shows the generated assembly code, which is highly compressed and uses placeholder values like 0x00 and 0x0000. The right side shows the resulting bytecode, represented as a long string of hex digits.

The rules of the game

- The owner of the Smart Contract is the creator
 - The owner writes the code
 - The owner sends a transaction to the blockchain with that code attached
 - The Smart Contract now has a unique address & balance
- What are the governance rules?
 - Every HelloToken costs 2 finney ($2/1000 \text{ ETH} = 2 \times 10^{15} \text{ wei}$)
 - Minimum threshold: 2 finney
 - If I want HelloTokens, I send ETH to the contract
 - If I send the contract 11 finney, I get 5 HelloTokens in my balance
 - If I send 1 finney, my request is refused

Your brand new token in 5 minutes or less

```
1 // SPDX-License-Identifier: CC-BY-SA-4.0
2 pragma solidity >=0.8.0 <0.9.0;
3
4 contract HelloToken {
5     address public minter;
6     mapping (address => uint) public balance;
7     uint public constant PRICE = 2 * 1e15;
8     // uint public constant PRICE = 2 finney;
9     // finney is no longer a supported denomination since Solidity v0.7.0
10
11 constructor() {
12     minter = msg.sender;
13 }
14
15 function mint() public payable {
16     require(msg.value >= PRICE, "Not enough value for a token!");
17     balance[msg.sender] += msg.value / PRICE;
18     // Guess guess, where does the remainder of the msg.value end?
19 }
20
21 function transfer(uint amount, address to) public {
22     require(balance[msg.sender] >= amount, "Not enough tokens!");
23     balance[msg.sender] -= amount;
24     balance[to] += amount;
25 }
26 }
```

The name of the contract and the token coincide: "HelloToken"

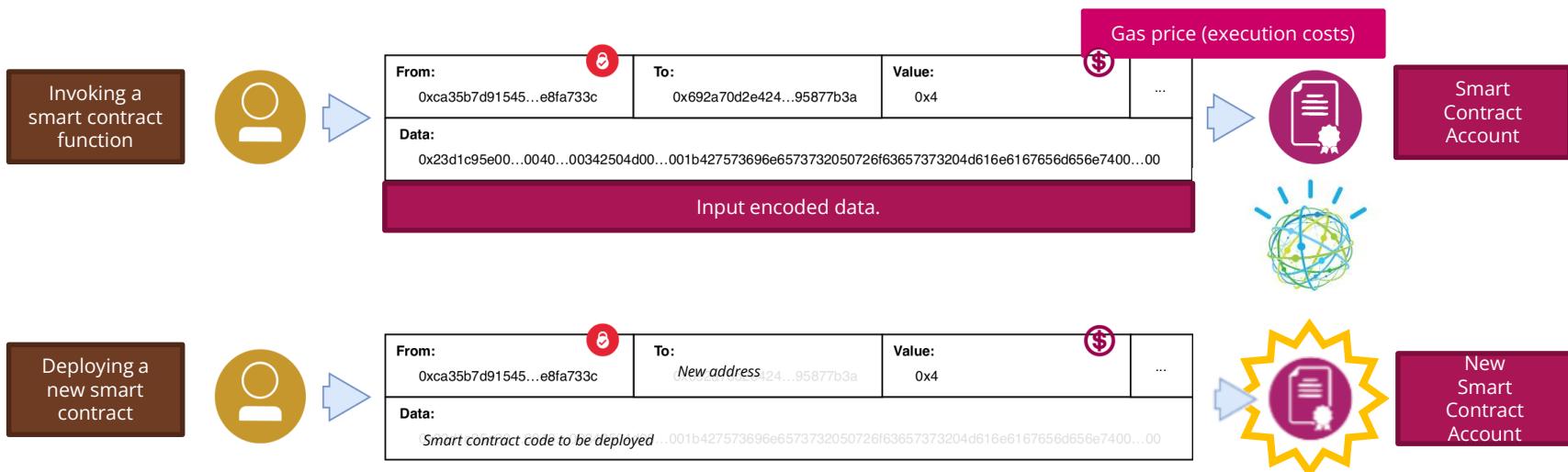
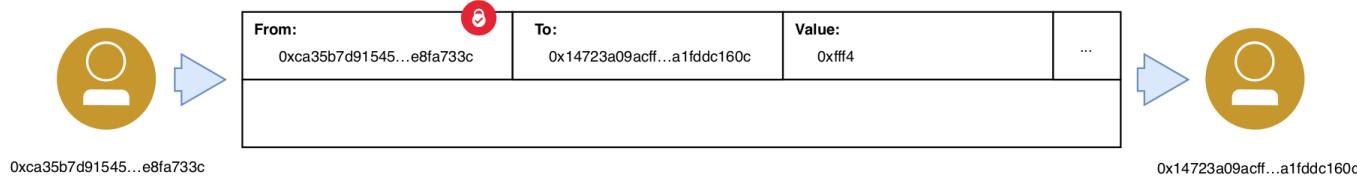
msg.sender is the identity (address) of the one who calls the function

msg.value is the amount of wei (a fraction of ETH) attached



Do not do this at home. This code is simple yet prone to basically any well-known attack.

A programmable distributed environment



The rules of the game (updated)

- The owner of the Smart Contract is the creator
 - The owner writes the code
 - The owner sends a transaction to the blockchain with that code attached
 - The Smart Contract now has a unique address & balance
- What are the governance rules?
 - Every HelloToken costs 2 finney (1/1000 ETH)
 - Minimum threshold: 2 finney
 - If I want HelloTokens, I send ETH to the contract
 - If I send the contract 11 finney, I get 5 HelloTokens in my balance
 - If I send 1 finney, my request is refused
- Who gets the ETH of the contract?
 - The ETH are stored in the Smart Contract account
 - The creator can get the ETH upon the self-destruction of the contract

Your brand new token in 5 minutes or less

```
1 // SPDX-License-Identifier: CC-BY-SA-4.0
2 pragma solidity >=0.8.0 <0.9.0;
3
4 contract HelloToken {
5     address public minter;
6     mapping (address => uint) public balance;
7     uint public constant PRICE = 2 * 1e15;
8     // uint public constant PRICE = 2 finney;
9     // finney is no longer a supported denomination since Solidity v.0.7.0
10
11 constructor() {
12     minter = msg.sender;
13 }
14
15 function mint() public payable {
16     require(msg.value >= PRICE, "Not enough value for a token!");
17     balance[msg.sender] += msg.value / PRICE;
18     // Guess guess, where does the remainder of the msg.value end?
19 }
20
21 function transfer(uint amount, address to) public {
22     require(balance[msg.sender] >= amount, "Not enough tokens!");
23     balance[msg.sender] -= amount;
24     balance[to] += amount;
25 }
26
27 function terminate() public {
28     require(msg.sender == minter, "You cannot terminate the contract!");
29     selfdestruct(payable(minter));
30 }
31 }
```

At self-destruction, the contract sends its belongings to the account that deployed it



Do not do this at home. This code is simple yet prone to basically any well-known attack.

Your brand new token in 5 minutes or less

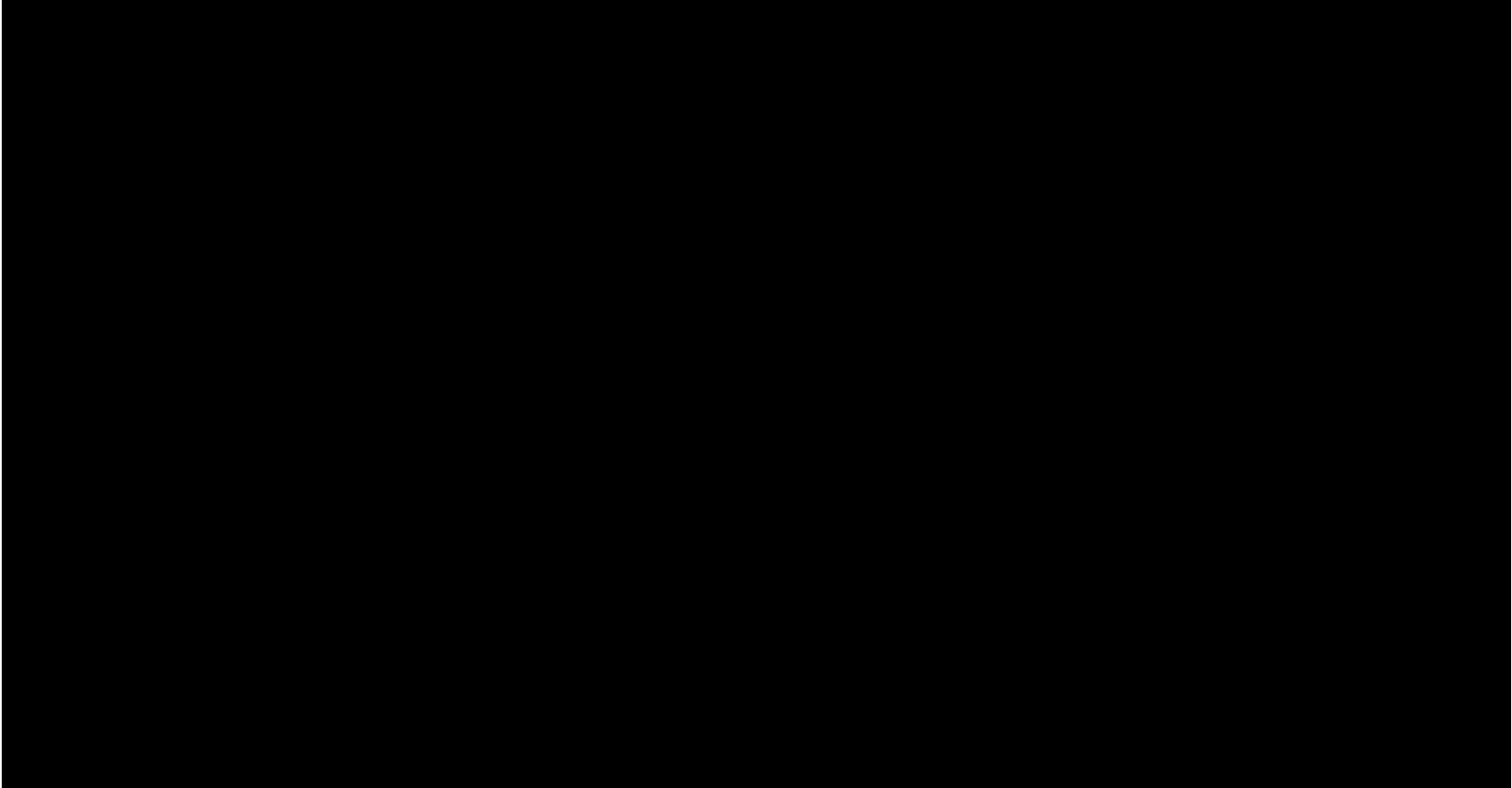
The screenshot illustrates the process of creating a new token in a few steps:

- Solidity Code:** The left side shows the `0-HelloToken.js`, `index.html`, and `HelloToken.sol` files. The `HelloToken.sol` file contains the Solidity code for the token contract.
- MetaMask Notification:** A central modal window titled "MetaMask Notification" displays a transaction for minting 0.005 tokens. It shows the amount in finney (0.005), the price (0.87 €), and the total cost (1.57 €). It includes tabs for "DETAILS" and "DATA".
- Web Browser:** On the right, a browser window titled "Hello Token!" shows the "Buy Hello Tokens!" page. It features a "Minting form" where the user can enter the number of tokens (5) and a "Transfer tokens" section where they can transfer tokens to another address (0x6e0a8c4Dc9deCa8946435BDD2738bD6eAb348FdA).



Utrecht
University

Announcement



Challenges in building Billion User Cloud Applications (Summer School, Venice, 6-11 October 2024)

- Team of instructors:
 - Dan Ardelean (Distinguished Software Engineer in Google Cloud)
 - Amer Diwan (Distinguished Software Engineer at Google),
 - JJ Furman (founder and creator of Megastore, the system behind Gmail and Drive)
- Organised by Alessandro Panconesi and Emanuele Panizzi (Sapienza University of Rome) for BICI
- Generous fellowships are provided
 - Full room and board for the duration of the school



The **summer school** will teach you about the challenges of building billion-user systems and present approaches to address those challenges.



Interested? Contact me at c.diciccio@uu.nl and Emanuele Panizzi at panizzi@di.uniroma1.it

Agenda for today



13:15 - 13:20:
Recap

13:20 - 14:00:
Smart Contracts and
DApps

14:15 - 15:00:
Tokens and hands on
Solidity

15:00 - 15:30:
Feedback

15:30 - 17:00:
Architectural debate



The information in this presentation has been compiled with the utmost care,
but no rights can be derived from its contents.