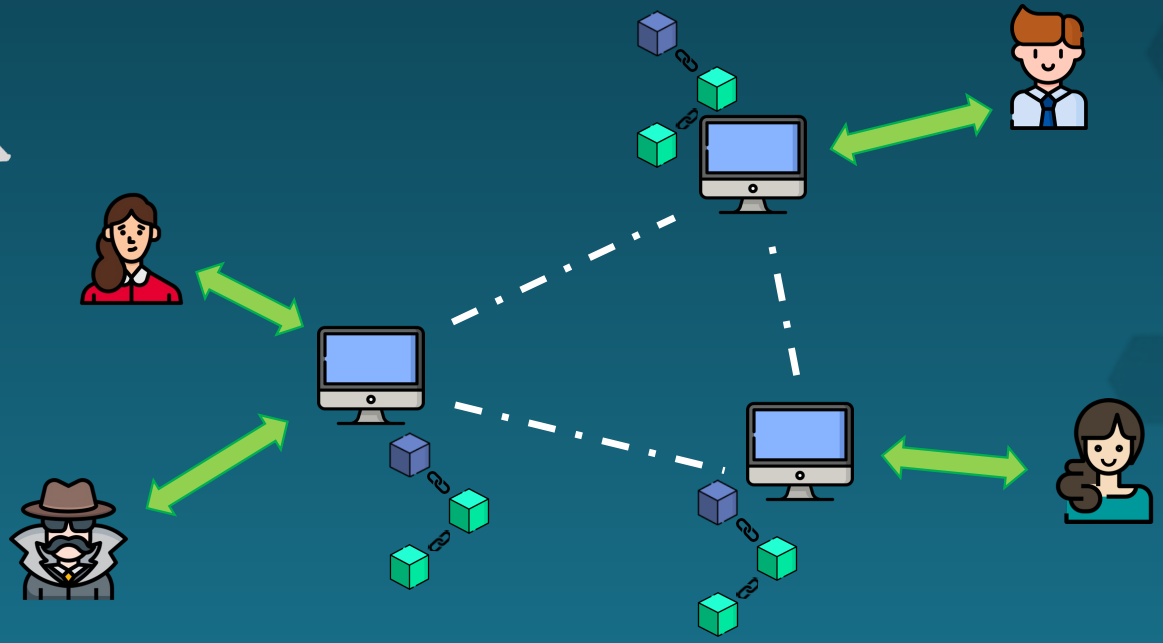
Blockchain and Applications

Chapter 5

Smart Contracts and DApps

Current state

We now have a blockchain that runs on a network and benefits all perks of a decentralized system



Bitcoin (2008)

Satoshi Nakamoto

Bitcoin: A Peer-to-Peer Electronic Cash System

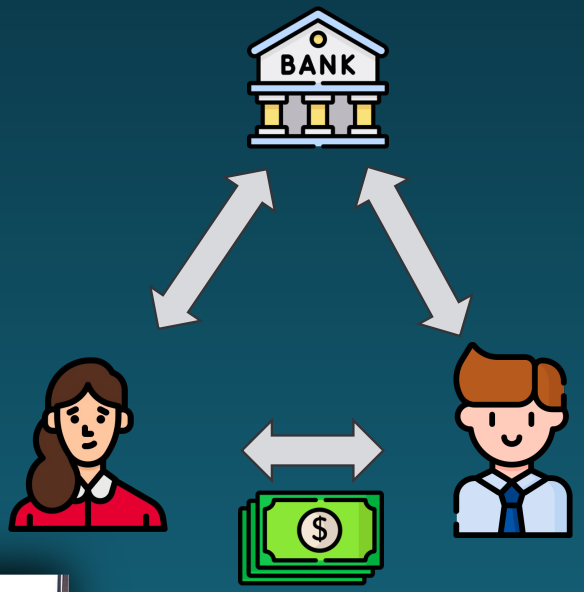
Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Designed for cash transfer

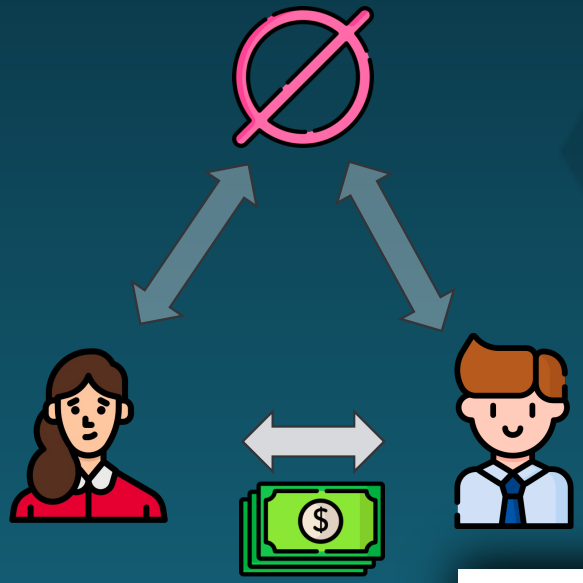
Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

It opens up a new world of possibility, due to two properties of blockchains...

1. Third party

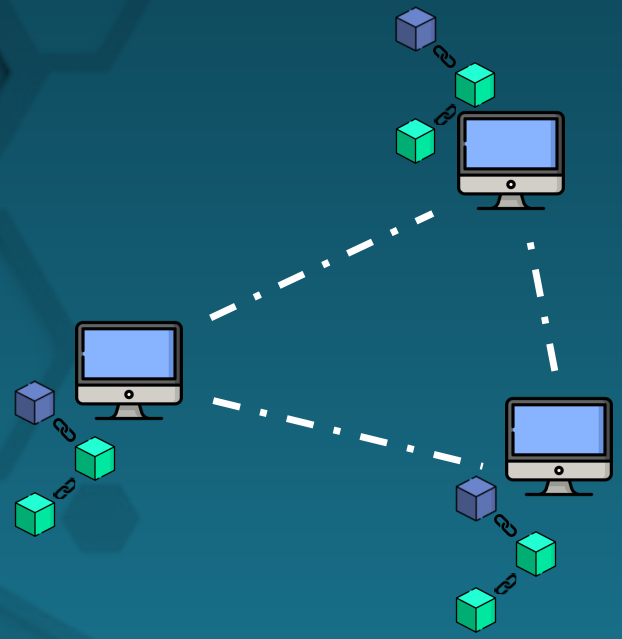


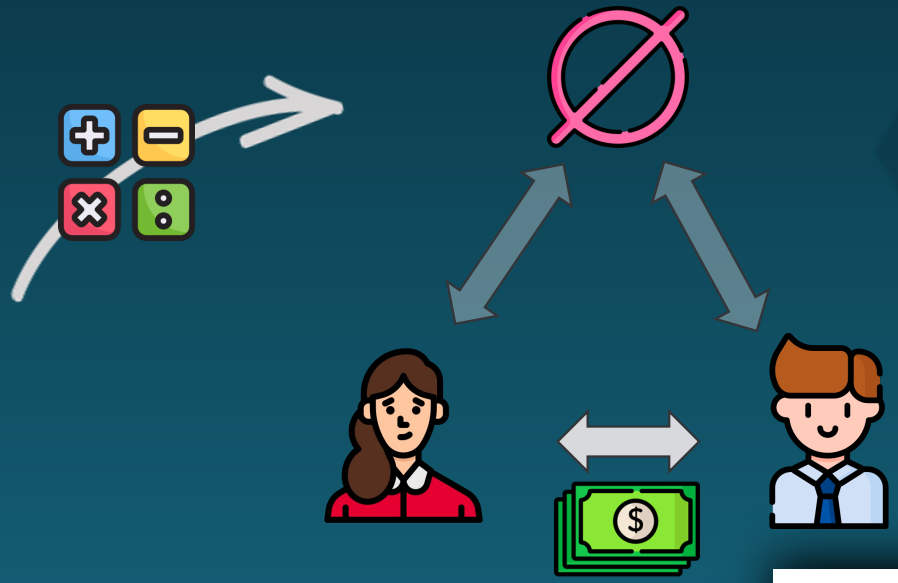
The bank still acts as a third party in the process



No third party is involved... or at least, it seems like

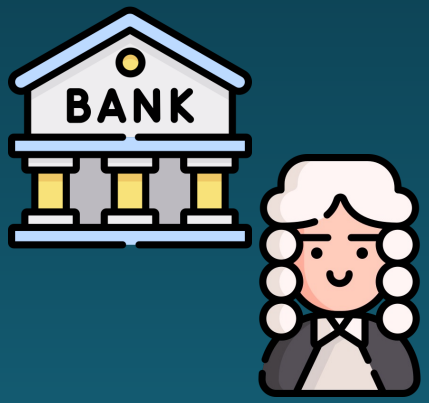
1. Third party





A set of mathematical rules (the blockchain) acts as a third party

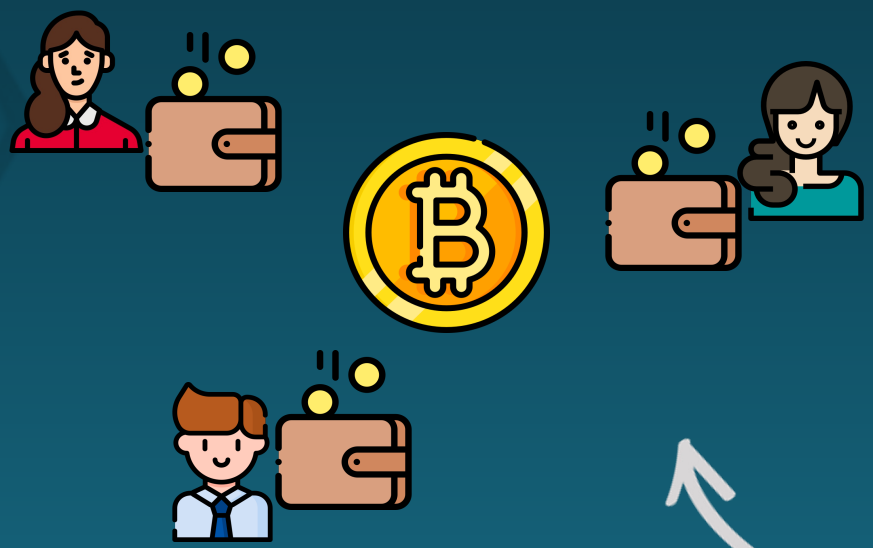
1. Third party



Mostly trusted but not flawless

Unbreakable and trustworthy

2. A world of statements

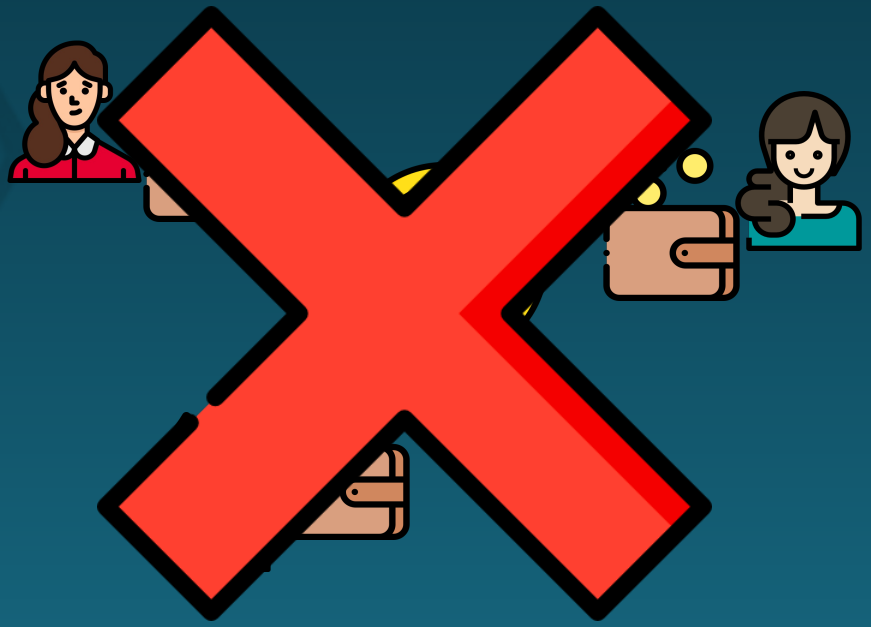


Bitcoin (or any cryptocurrency)

=

I hold tokens inside my wallet, like a physical fiat currency like euros or dollars

2. A world of statements



Doesn't work this way at all !!

2. A world of statements

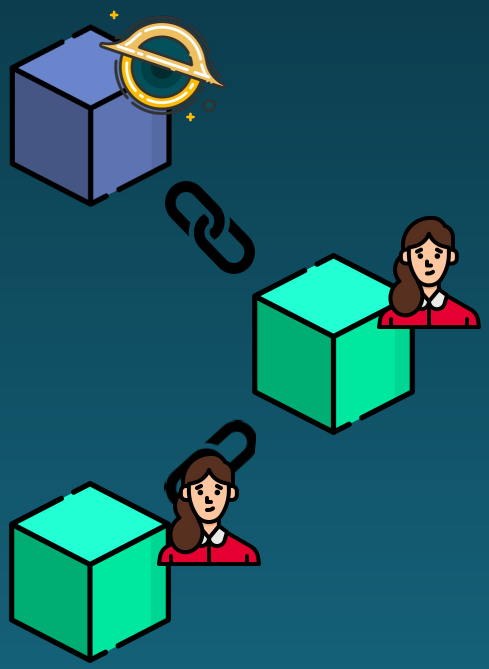
I give 2 bamboozloos to Bob

-2

+2

A statement acts like a bank check : it is a money transfer

2. A world of statements



From	To	Amount
Alice	Bob	0.2
Alice	Charlie	0.3
Charlie	Delphine	0.1
Bob	Delphine	0.2
…	…	…

The ledger keeps track of all transactions, in a very strict and universal manner : it is like an algorithm

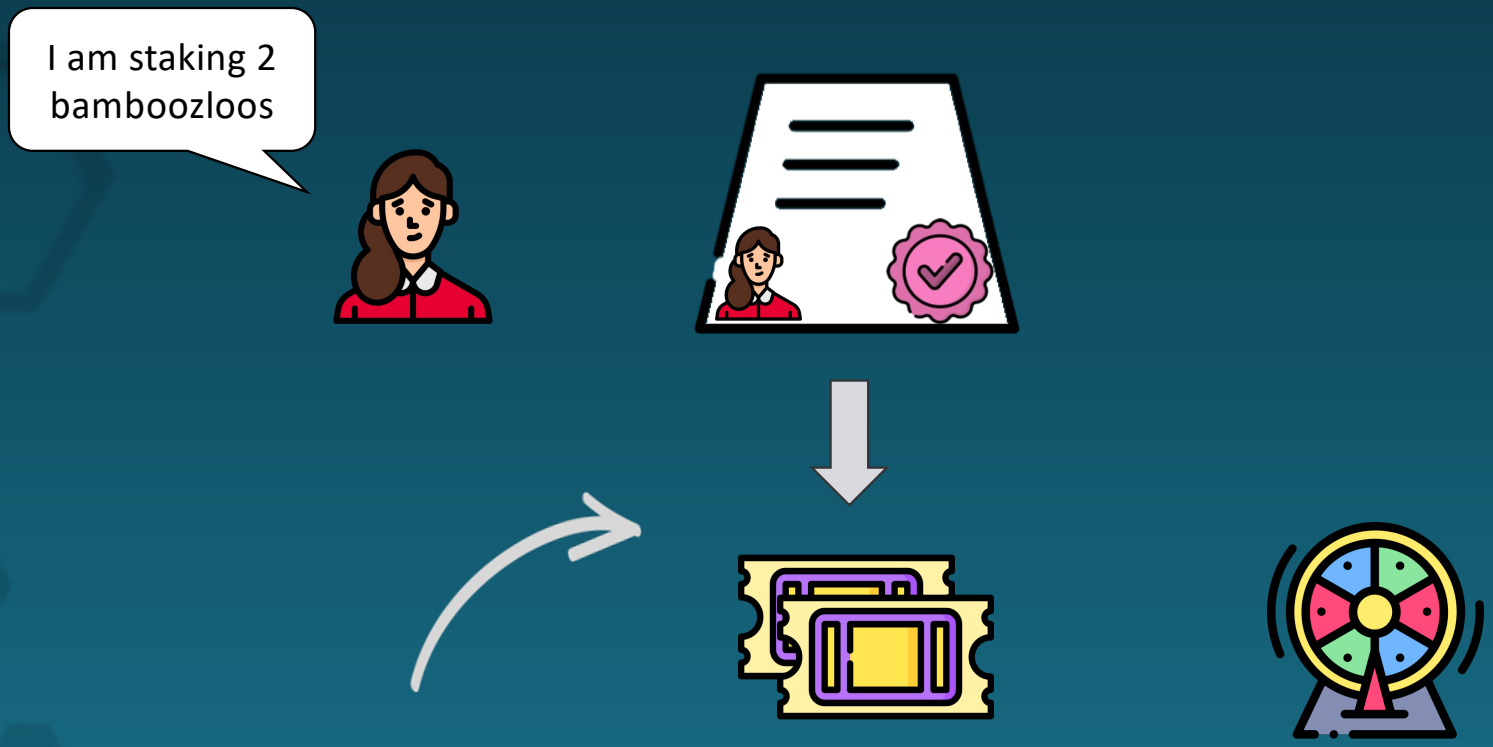
 1.5

 0

 0.2

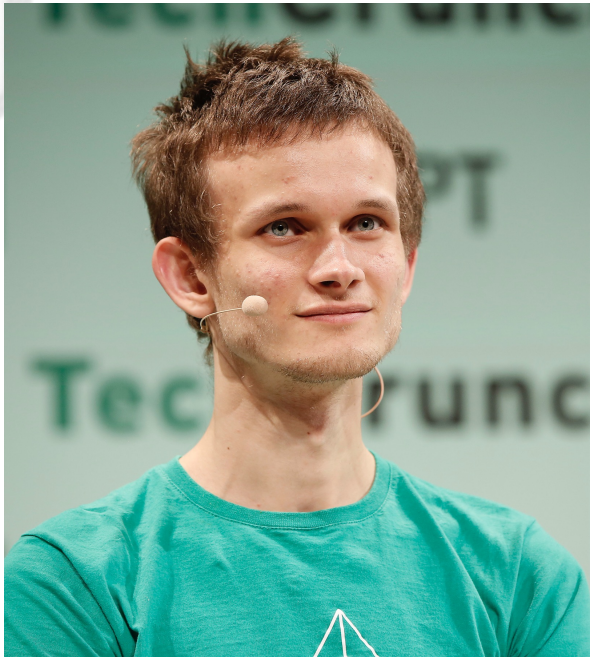
 0.3

2. A world of statements



It has been decided beforehand that X staked tokens = X tickets for the next raffles

Ethereum (2015)

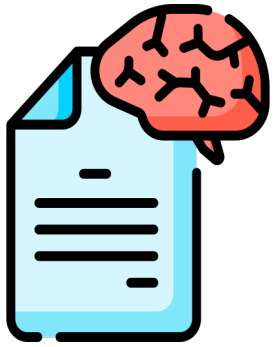


Vitalik Buterin

A blockchain is just one computer program that runs on a decentralized network

Ethereum (2015)

Base for a good blockchain : a cryptocurrency

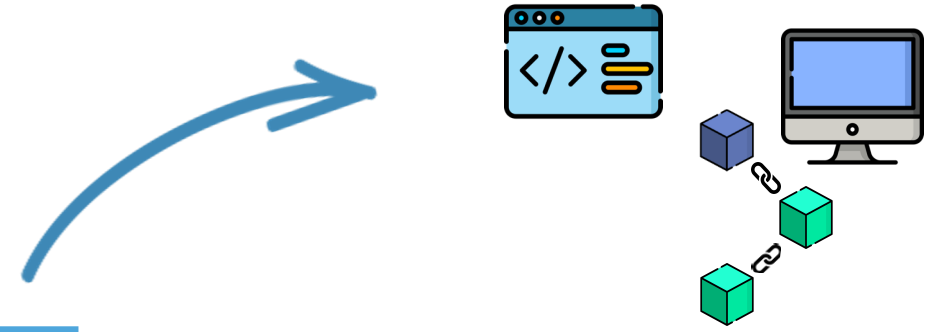


BUT ! Also let users deploy "smart contracts"

Bitcoin lacks this part

What is a Smart Contract ?

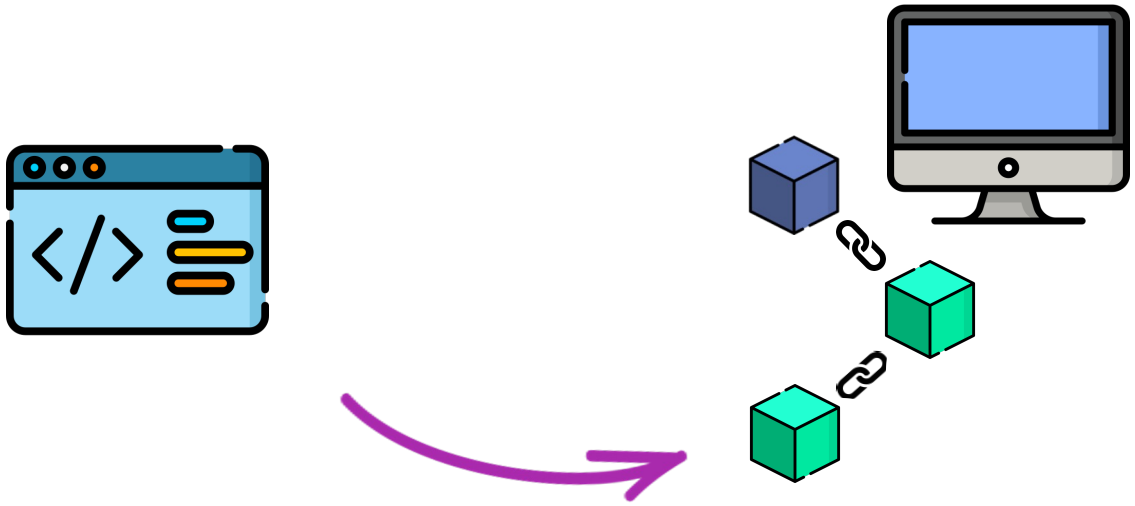
From	To	Amount
Alice	Bob	0.2
Alice	Charlie	0.3
Charlie	Delphine	0.1
Bob	Delphine	0.2
…	…	…



It is the official blockchain software that contains the algorithm to calculate the balance of a user

Delphine owns 0.3

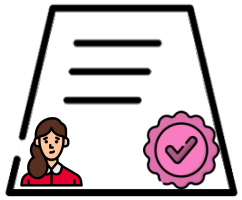
What is a Smart Contract ?



What if the piece of algorithm used to control a system was stored inside the blockchain directly ?

Smart Contracts

I want to rent my appartment



Terms of the contract

- Both parties can retreat at any time

- The tenant is automatically taken 1 bamboozloo every month

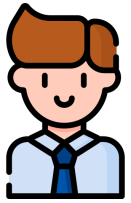
- Any raise in rent should happen, both parties need to agree

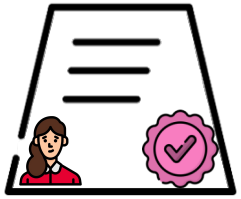
- ...

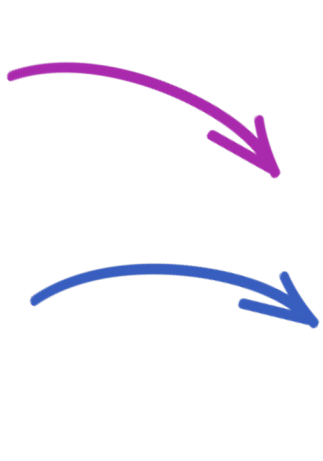
Smart Contracts

I want to put my apartment for rent

I want to rent Alice's apartment starting March 2024







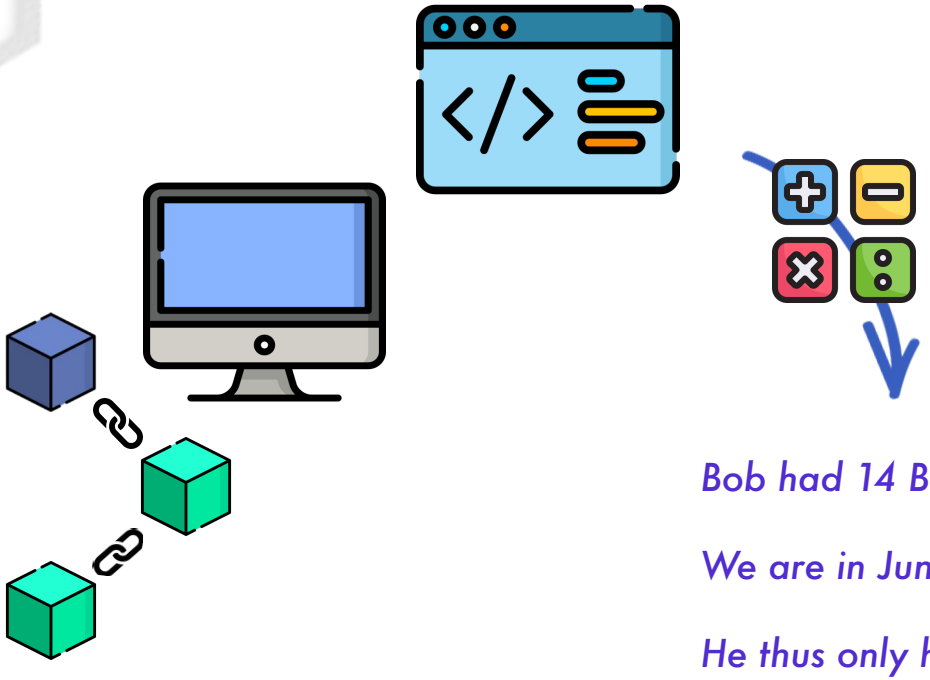
March 2024

-1

April 2024

-1

Smart Contracts



The laws of mathematics and algorithmics act as third party to ensure payment is enforced

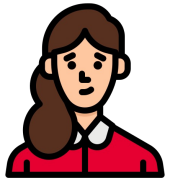
Bob had 14 Bamboozloos in February 2024.

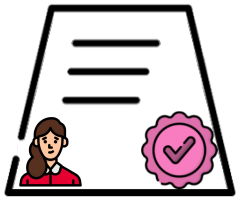
We are in June 2024.

He thus only has 10 Bamboozloos left, as per the terms of the rental contract

Data storage in Smart Contracts

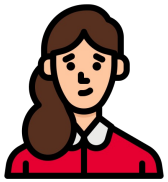
I want to register a new association for my school

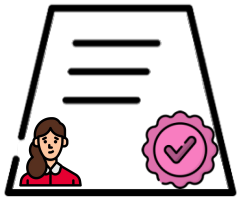




Members = { }

I register to my association



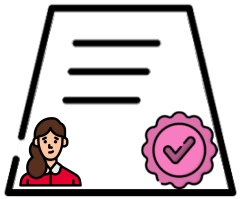


Members = { Alice }

I register to Alice's association

Members = { Alice, Bob }

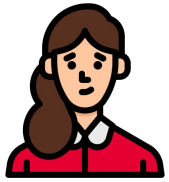
I am quitting my association

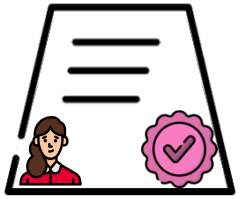


Members = { Bob }

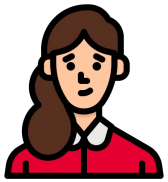
Data storage in Smart Contracts

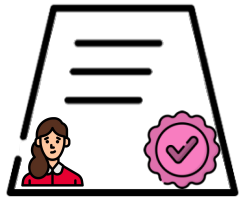
I want to register a new association for my school





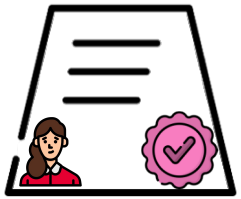
I register to my association





I register to Alice's association

I am quitting my association



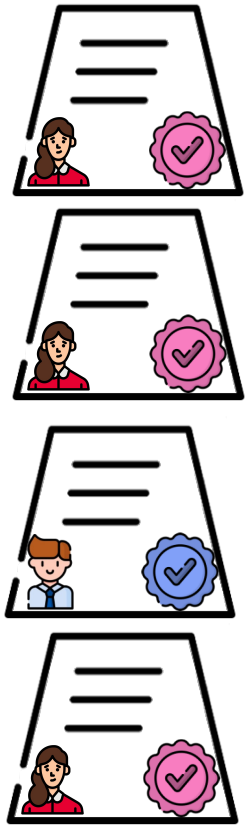
Data is never really "gone", it is just updated.

Data persistence is a high stake

Members = { Bob }

Reading Smart Contracts

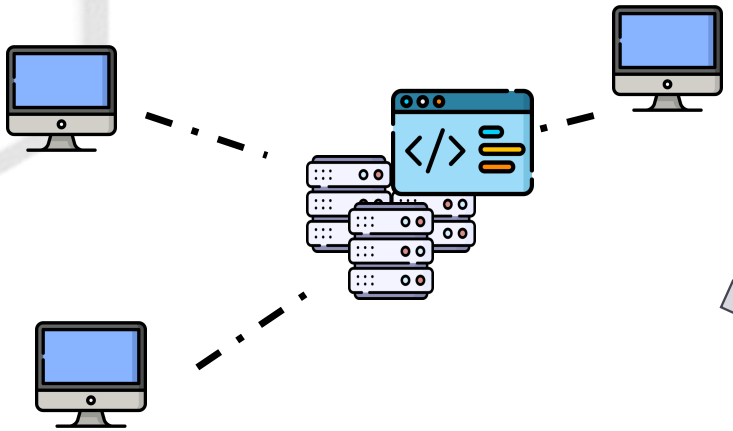
Who is member of this association ?



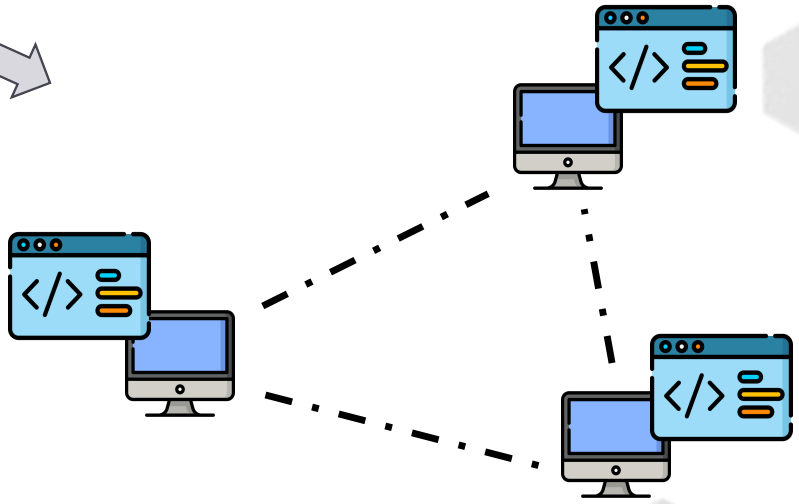
Delphine goes from Smart Contract declaration all the way down to last certificate and locally applies changes to the contract

Members = { Bob }

From Smart Contracts to DApps



Blockchains allow a set of rules to be enforced without having a central authority

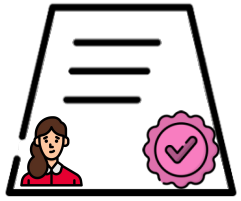


It more generally allows developers to write decentralized applications (DApps) that run autonomously on the whole network

Validators

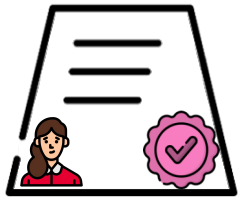
Validators run every smart contract locally, like an independent software, to validate certificates related to this smart contract

I validate Alice's departure, since she was indeed in the member list



“Decentralized” applications ?

I want to rent my appartment



Every smart contract has an owner

Alice owns the source code for her rental. She can hard code pretty much anything she wants

Isn’t that centralized ?

A new expertise : Smart Contract audit

Companies specialized in reviewing decentralized applications for any security flaw

Example : BAYC (Etherscan)

ETH Price: $1,677.29 (-2.05%) Gas: 44 Gwei

Search by Address / Txn Hash / Block / Token / Domain Name

Contract Source Code Verified (Exact Match)

Contract Name:	**BoredApeYachtClub**	Optimization Enabled:	**No** with **200** runs
Compiler Version	**v0.7.0+commit.9e61f92b**	Other Settings:	**default** evmVersion, **MIT** license

Contract Source Code (Solidity)

```
66  /**
67   * @dev Required interface of an ERC721 compliant contract.
68   */
69  interface IERC721 is IERC165 {
70      /**
71       * @dev Emitted when `tokenId` token is transferred from `from` to `to`.
72       */
73      event Transfer(address indexed from, address indexed to, uint256 indexed tokenId);
74
75      /**
76       * @dev Emitted when `owner` enables `approved` to manage the `tokenId` token.
77       */
78      event Approval(address indexed owner, address indexed approved, uint256 indexed tokenId);
79
80      /**
81       * @dev Emitted when `owner` enables or disables (`approved`) `operator` to manage all of its assets.
82       */
83      event ApprovalForAll(address indexed owner, address indexed operator, bool approved);
84
85      /**
86       * @dev Returns the number of tokens in ``owner``'s account.
87       */
88      function balanceOf(address owner) external view returns (uint256 balance);
89
90      /**
```

NFTs

Non Fungible Tokens

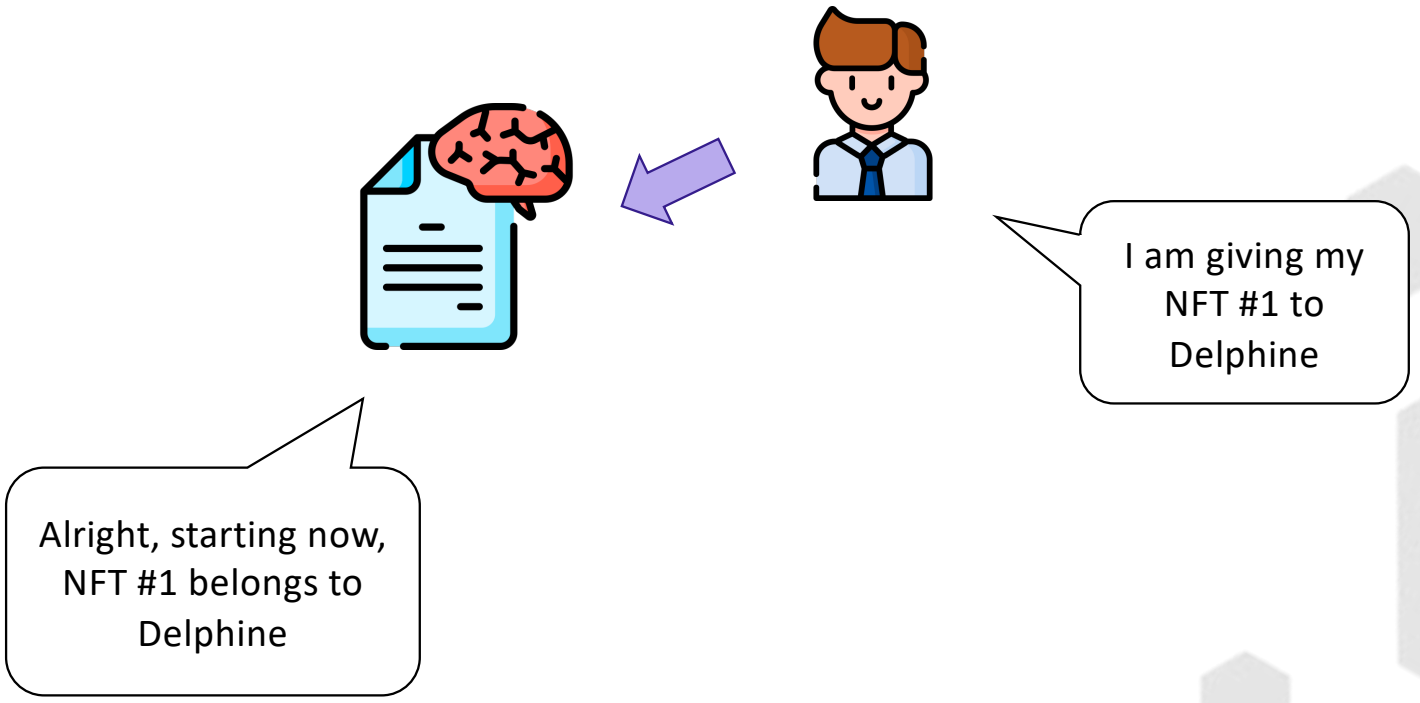
I create a collection of 10000 NFTs

I am minting the next NFT

You get NFT #1

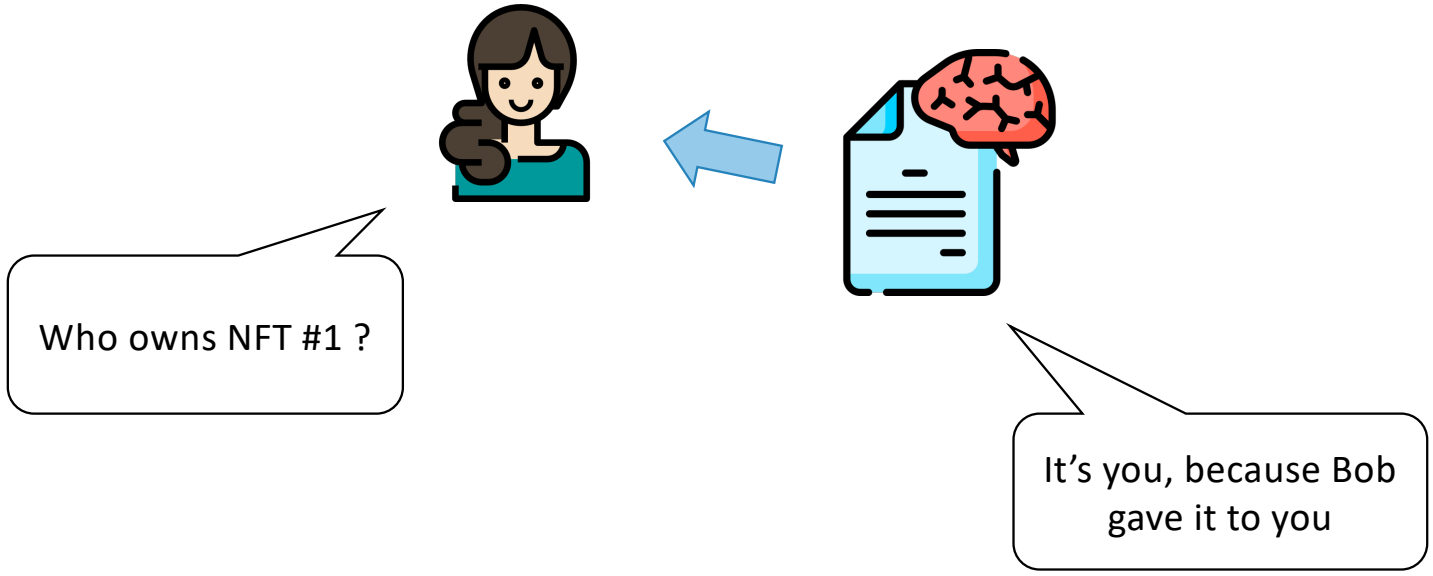
NFTs

Non Fungible Tokens



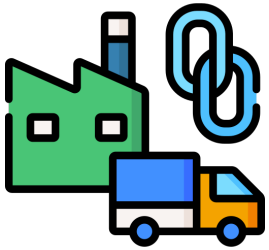
NFTs

Non Fungible Tokens

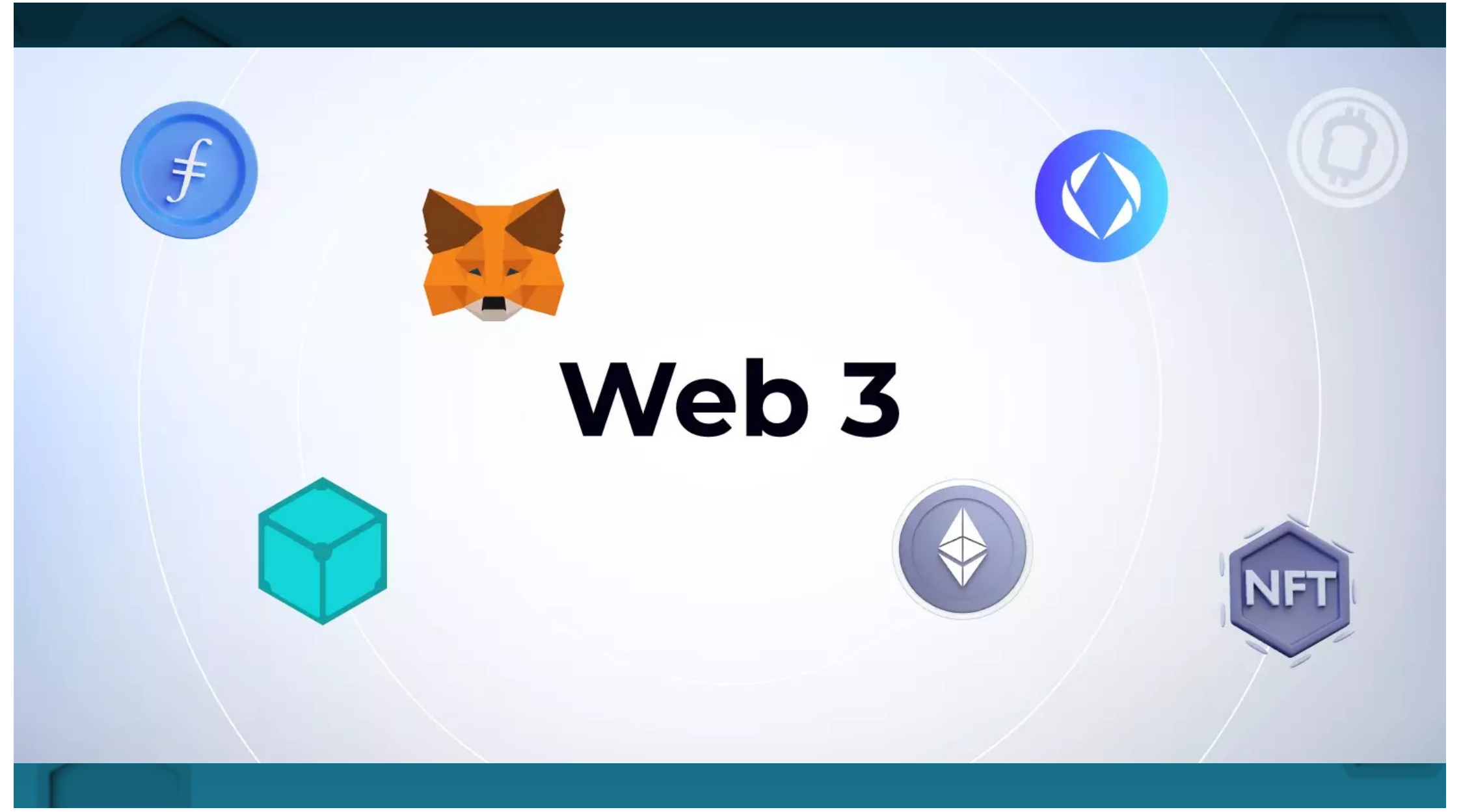


NFTs

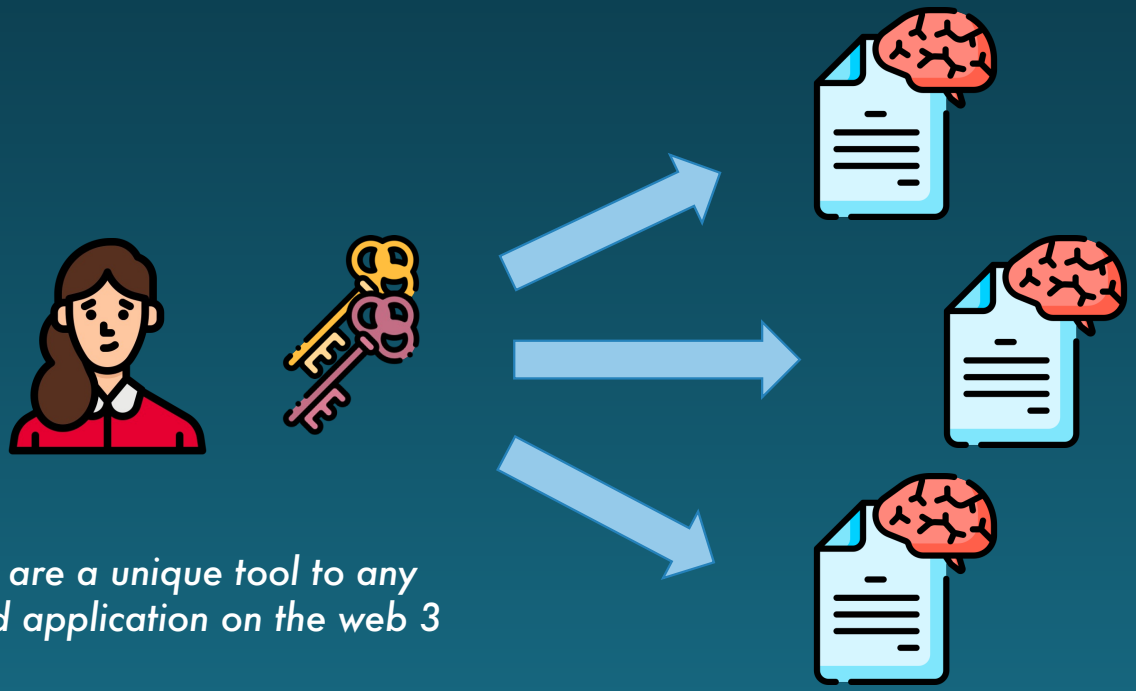
Non Fungible Tokens



One NFT can translate to literally ANYTHING

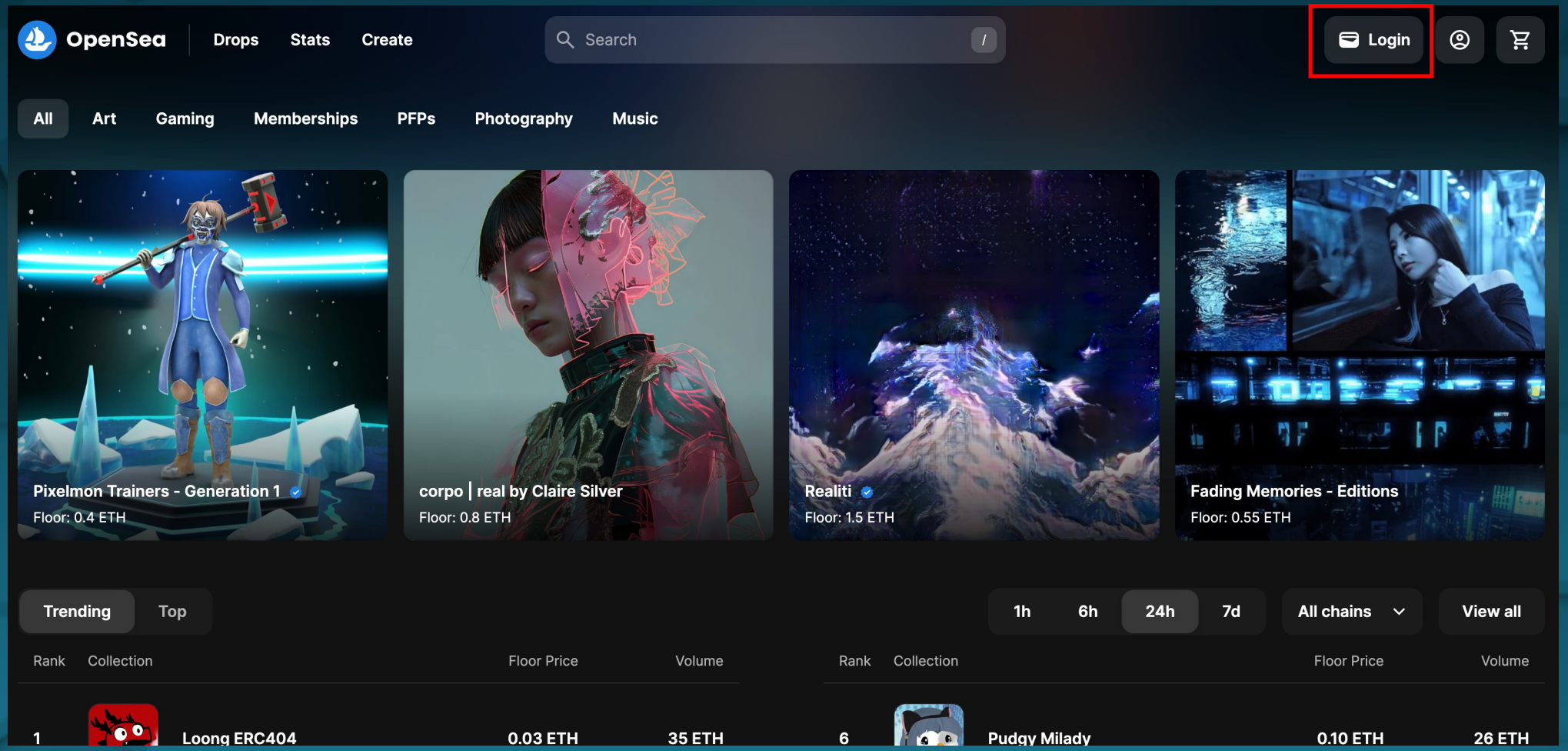

Web 3
NFT

Identity on Web3

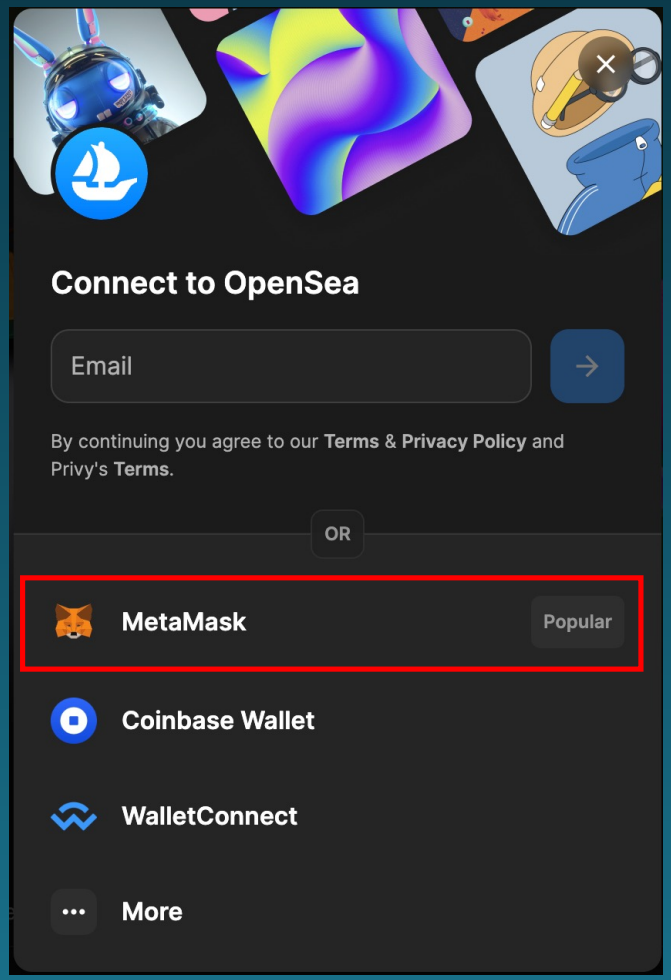


Alice's keys are a unique tool to any decentralized application on the web 3

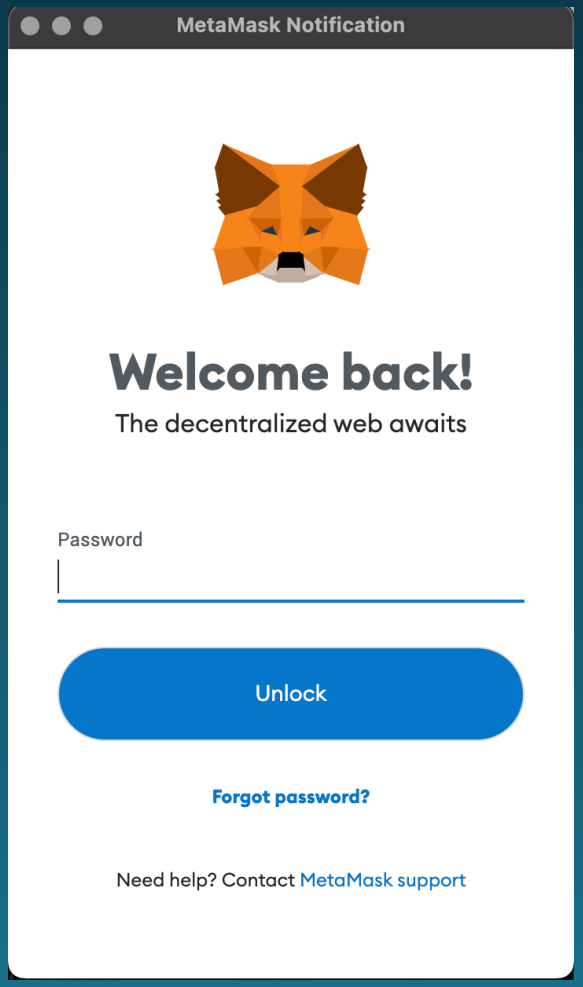
Web3 – a use case



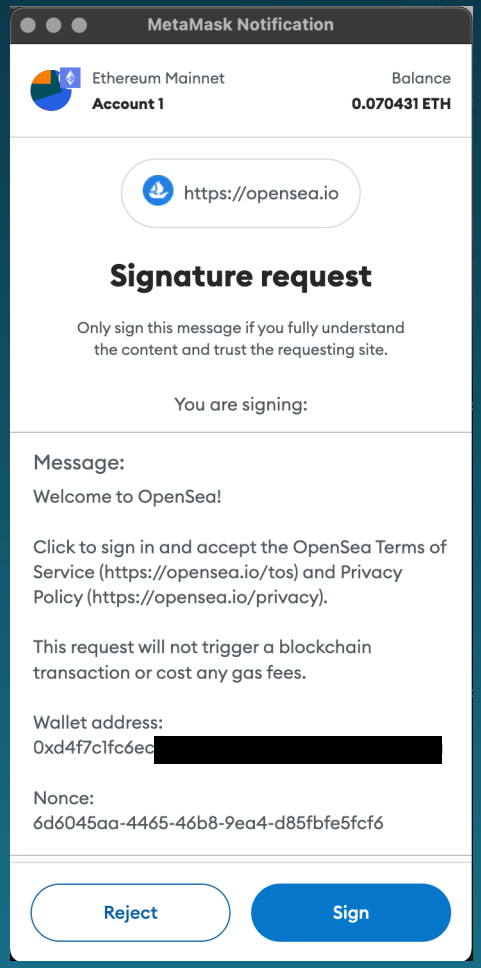
Web3 – a use case



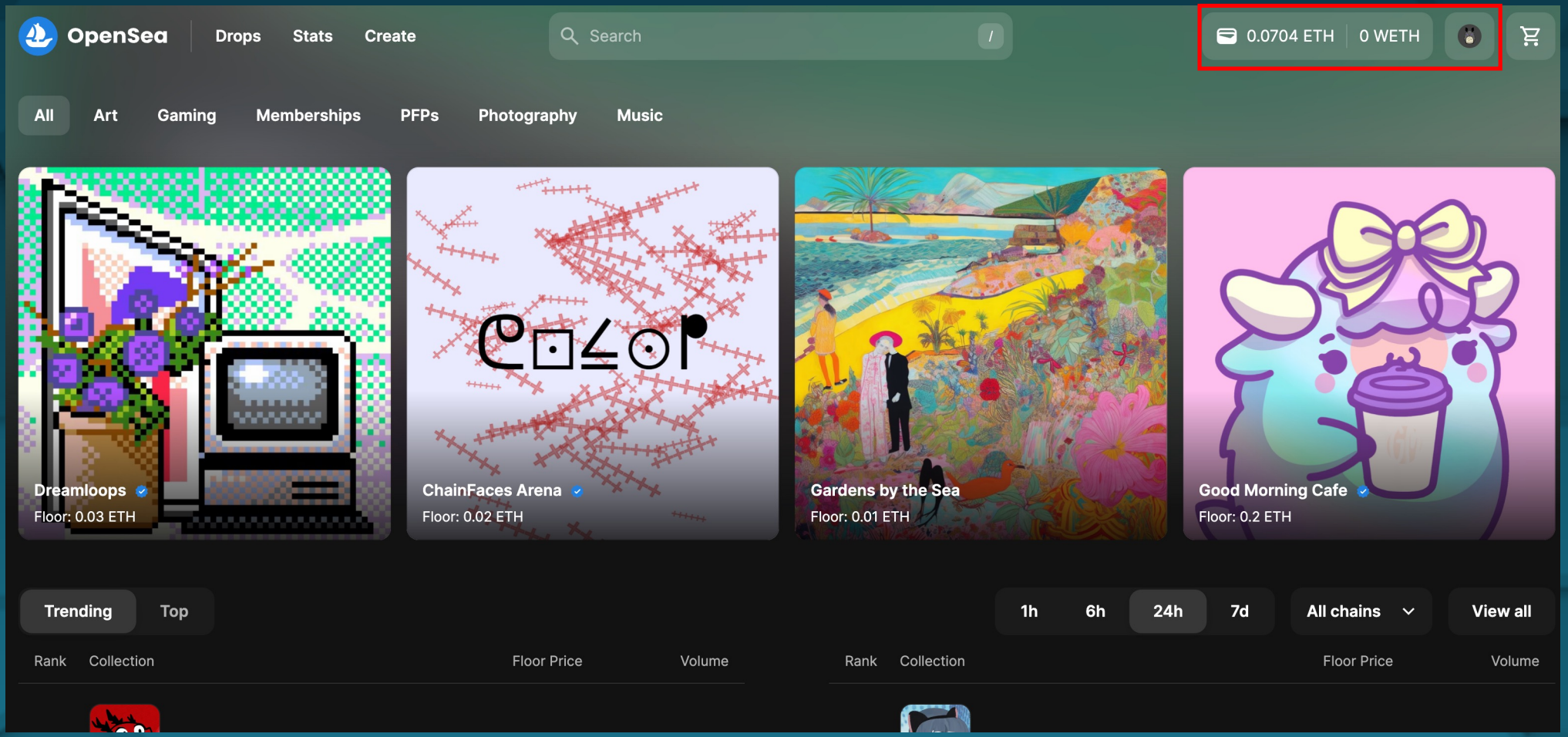
Web3 – a use case



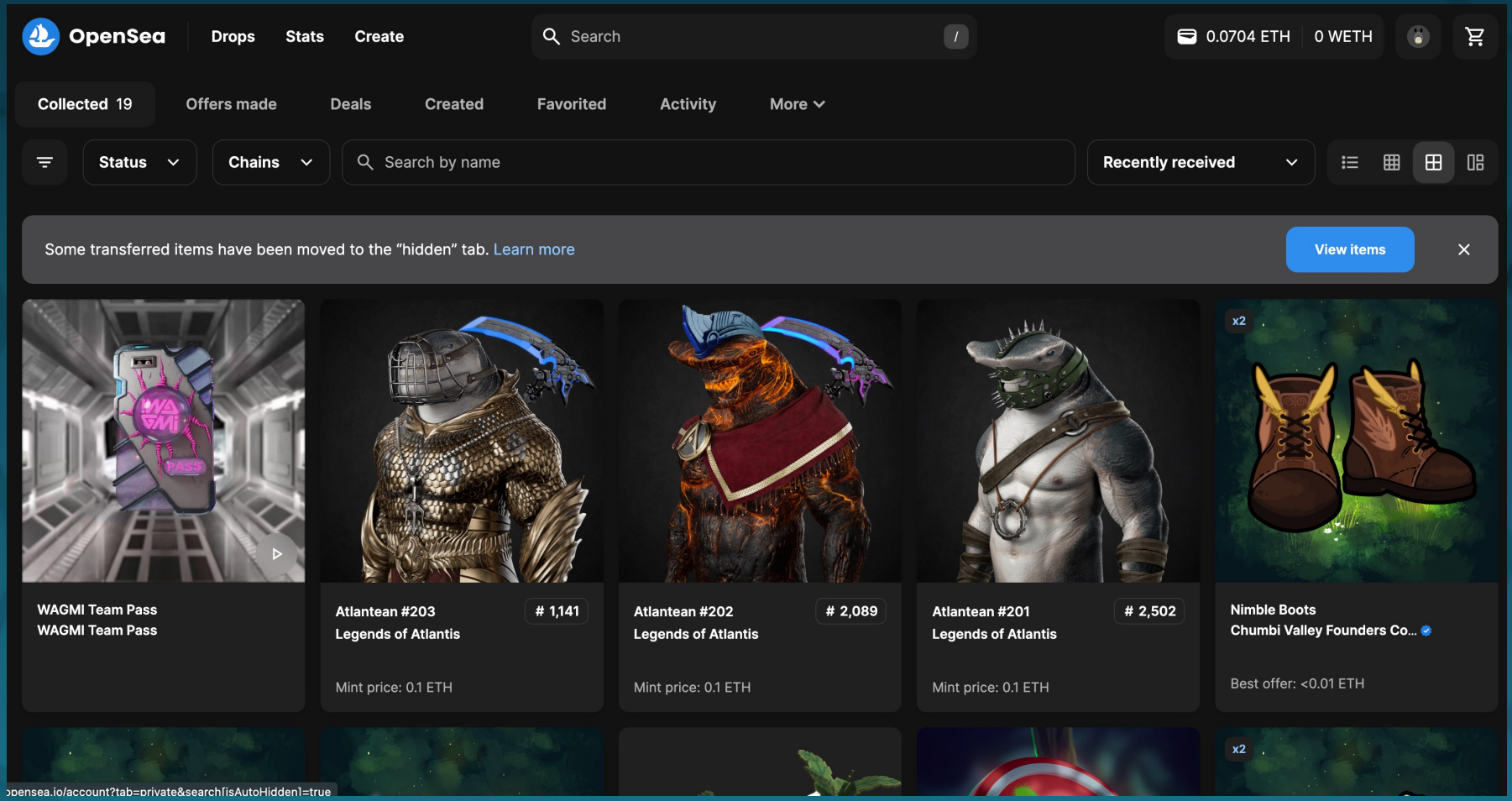
Web3 – a use case



Web3 – a use case



Web3 – a use case



Solidity

Ethereum's Smart Contract language

Solidity – 2014

Gavin Wood – Ethereum co-founder

"One computer for the entire planet"

Solidity – OOP smart contract language
August 2014

Solidity – 2014

```
pragma solidity >=0.4.16 <0.9.0;

contract SimpleStorage {

    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

    function get() public view returns (uint) {
        return storedData;
    }

}
```

Like a class definition

22498 gas

Gas = energy consumption by validators to run the code

public = anyone can call it

2437 gas

view = contract is not modified

Solidity – Owner

```
pragma solidity >=0.4.16 <0.9.0;

contract MyAssociation {

    struct Member {
        uint id;
        string firstName;
        string lastName;
    }

    uint256 public memberCount = 0;
    mapping(uint => Member) public members;

    address president;

    constructor() {    514390 gas 484600 gas
        president = msg.sender;
    }

    function incrementCount() internal {    infinite gas
        memberCount += 1;
    }

    function addMember(string memory firstName, string memory lastName) public {    infinite gas
        require(msg.sender == president);
        incrementCount();
        members[memberCount] = Member(memberCount, firstName, lastName);
    }

}
```

The caller of the constructor is the owner of the contract

« msg » contains info about call context

Will fail if the wrong user calls this

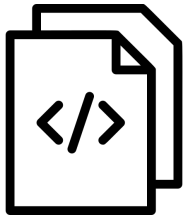
Solidity – Permissions

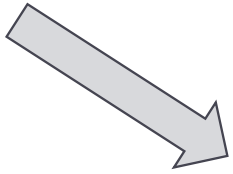
```
modifier onlyPresident() {
    require(msg.sender == president);
    _;
}

function addMember(string memory firstName, string memory lastName) public onlyPresident {
    incrementCount();
    members[memberCount] = Member(memberCount, firstName, lastName);
}
```

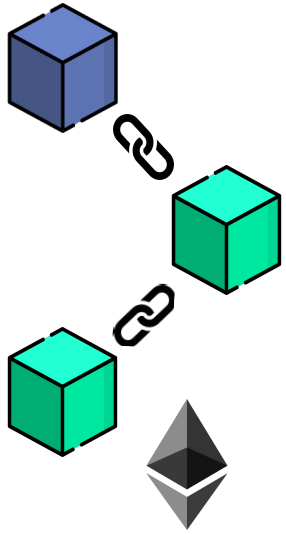
Permission granted to the president only

Special permission for this function

Solidity – Deployment



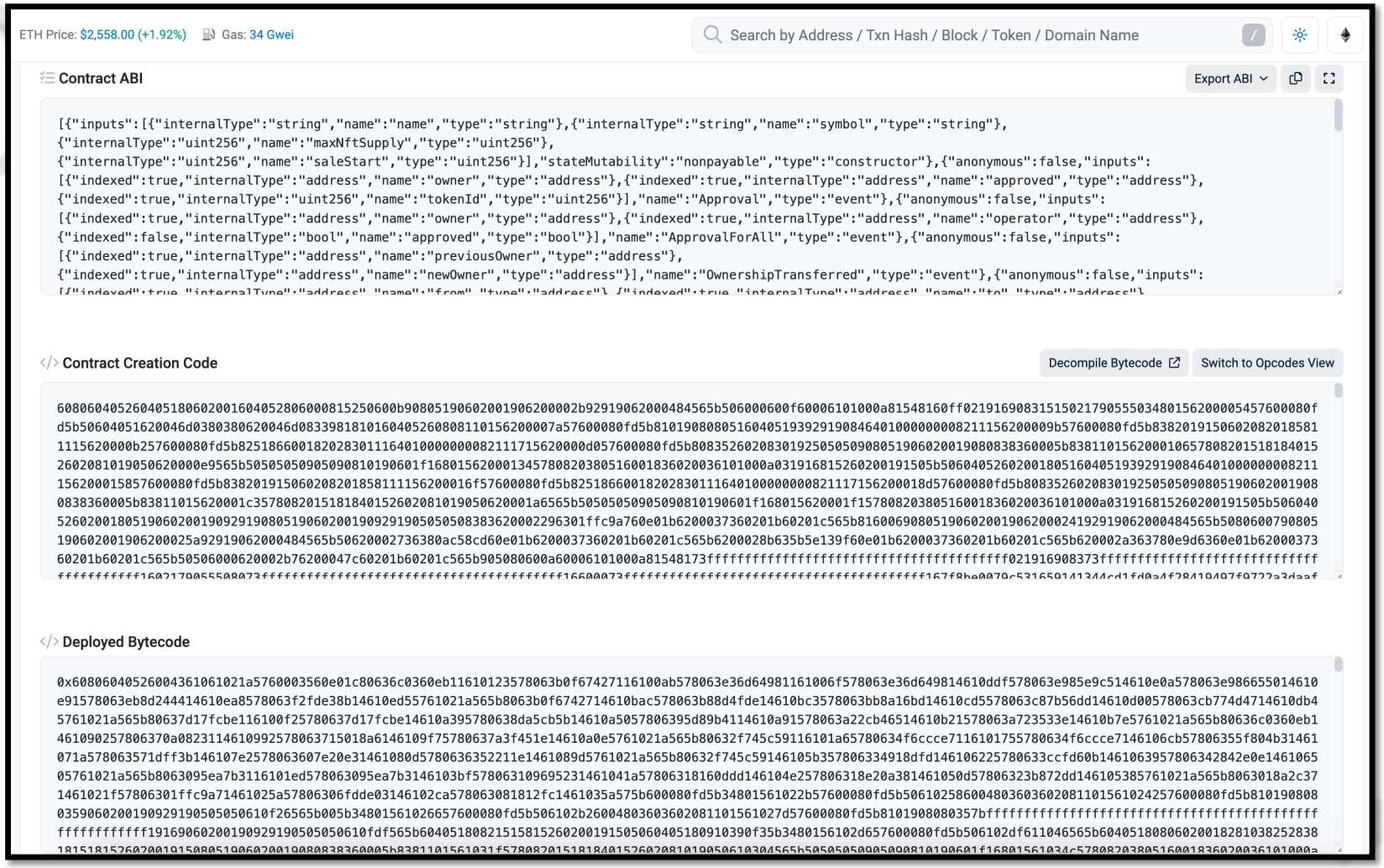




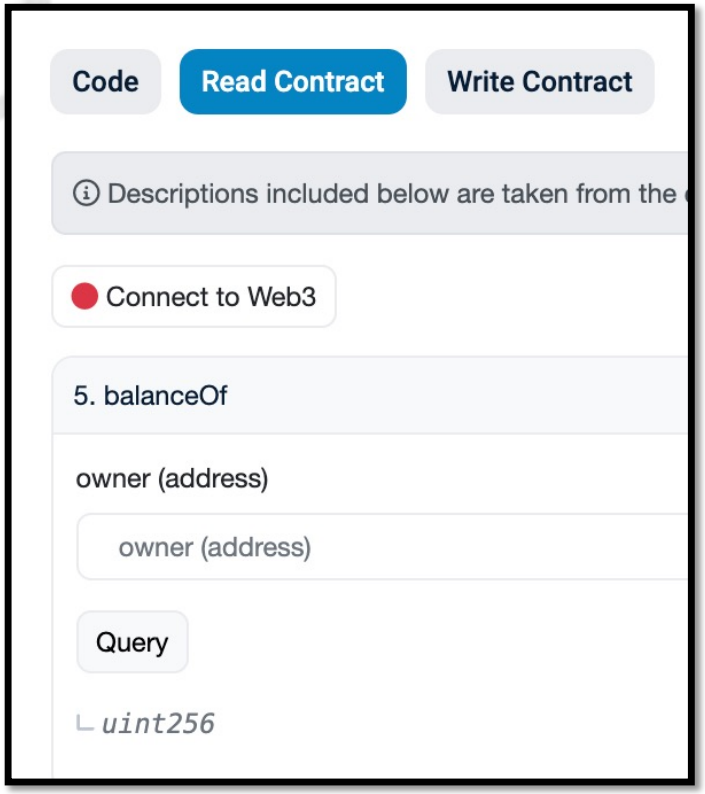
Code is compiled to binary using the official compiler (solc) or any framework (like Remix)

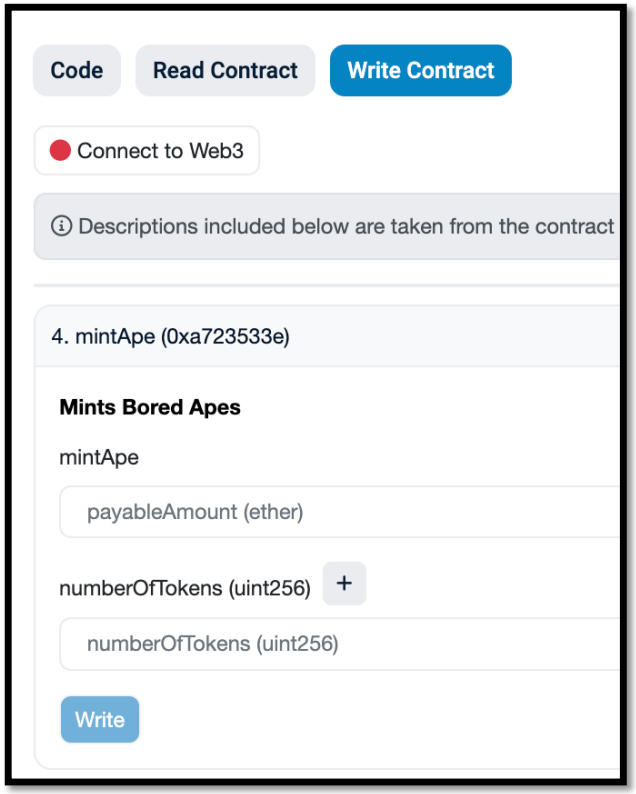
Binary code is written to the blockchain using a signed certificate, and by paying gas

Solidity – Deployed



Solidity – Interaction





Blockchain and Applications

Quiz 5

Smart Contracts and DApps