ERC 721 Token and info:

Intro and Collecting things:

We can emulate rare, collectible items with Ethereum tokens, and each of these tokens follows a novel standard in the Ethereum community known as ERC721. Ethereum Request for Comments 721, or ERC721, is an Ethereum Improvement Proposal introduced by Dieter Shirley in late 2017. It's a proposed* standard that would allow smart contracts to operate as tradeable tokens similar to ERC20. ERC721 tokens are unique in that the tokens are *non-fungible*.

On fungibility:

Fungible—being something (such as money or a commodity) of such a nature that one part or quantity may be replaced by another equal part or quantity in paying a debt or settling an account. Source: Merriam-Webster

Fungibility is, essentially, a characteristic of an asset, or token in this case, that determines whether items or quantities of the same or similar type can be completely interchangeable during exchange or utility. For example, the US Five Dollar bill below can be used to purchase a soda from a convenience store.

ERC721 tokens can be used in any exchange, but their value is a result of the uniqueness and rareness associated with each token. The standard defines the functions name , symbol , totalSupply , balanceOf , ownerOf , approve , takeOwnership , transfer , tokenOfOwnerByIndex , and tokenMetadata . It also defines two events: Transfer and Approval .

ERC721 defines a few functions that give it some compliance with the ERC20 token standard. It does this to make it easier for existing wallets to display simple information about the token. These functions let the smart contract that fits this standard act like a common cryptocurrency such as Bitcoin or Ethereum by defining functions that let users perform actions such as sending tokens to others and checking balances of accounts.contract

This function is used to tell outside contracts and applications the name of this token. An example implementation of the function can be as follows.

```
MyNFT {

  function name() constant returns (string name){
    return "My Non-Fungible Token";
  }
}
```

All functions above from:

Token definition:

Each token is it's own mini-database of who owns what. The "token" is just an entry in the token contract, and who "owns" a token is recorded in the contract. A token is never in "your" wallet.

Tokens are interfaces

An interface defines WHAT something does, not how it does it. ERC-20 is an interface. An ERC-20 token can have the interface and still do basically nothing. This can't be said enough: Having an ERC-20 token is MEANINGLESS. If someone asks for money for their ERC-20 token and the best reason they can do is that they have an Eth token, ignore them.

More advanced Tokens

ERC-721 is a more advanced token that is "non-fungible". Think of ERC-20 as the token type for things that are money (any $5 bill is worth the same as any other $5 bill, usually) and ERC-721 as the token type for collectibles (it's the Ethereum equivalent of baseball cards).

In ERC-721, each token is completely unique and non-interchangeable with other tokens. Another example: Pets. Many people have dogs, but THAT dog is theirs and they will not accept some other dog as a substitute. You can use ERC-721 tokens to represent those dogs and dog ownership.

——-

Aunyks/erc721- definitions.sol

```
pragma solidity ^0.4.18;

/**
 * @title ERC721 Non-Fungible Token Standard basic interface
 * @dev see https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md
 */
contract ERC721Basic {
  event Transfer(address indexed _from, address indexed _to, uint256 _tokenId);
  event Approval(address indexed _owner, address indexed _approved, uint256
_tokenId);
  event ApprovalForAll(address indexed _owner, address indexed _operator, bool
_approved);

  function balanceOf(address _owner) public view returns (uint256 _balance);
  function ownerOf(uint256 _tokenId) public view returns (address _owner);
  function exists(uint256 _tokenId) public view returns (bool _exists);

  function approve(address _to, uint256 _tokenId) public;
  function getApproved(uint256 _tokenId) public view returns (address _operator);

  function setApprovalForAll(address _operator, bool _approved) public;
  function isApprovedForAll(address _owner, address _operator) public view returns
(bool);

  function transferFrom(address _from, address _to, uint256 _tokenId) public;
  function safeTransferFrom(address _from, address _to, uint256 _tokenId) public;
  function safeTransferFrom(address _from, address _to, uint256 _tokenId, bytes _data)
public;
}
```

ERC-20: For money and money-like tokens.
ERC-721: For things and thing-like tokens.

What is ERC-721—

ERC-721 is a free, open standard that describes how to build non-fungible or unique tokens on the Ethereum blockchain. While most tokens are fungible (every token is the same as every other token), ERC-721 tokens are all unique.

Think of them like rare, one-of-a-kind collectables.

The Standard

ERC-721 defines a minimum interface a smart contract must implement to allow unique tokens to be managed, owned, and traded. It does not mandate a standard for token metadata or restrict adding supplemental functions.

```
pragma solidity ^0.4.20;

    /// @title ERC-721 Non-Fungible Token Standard
    /// @dev See https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md
    ///  Note: the ERC-165 identifier for this interface is 0x80ac58cd
    interface ERC721 /* is ERC165 */ {
      /// @dev This emits when ownership of any NFT changes by any mechanism.
      ///  This event emits when NFTs are created (`from` == 0) and destroyed
      ///  (`to` == 0). Exception: during contract creation, any number of NFTs
      ///  may be created and assigned without emitting Transfer. At the time of
      ///  any transfer, the approved address for that NFT (if any) is reset to none.
      event Transfer(address indexed _from, address indexed _to, uint256 indexed
_tokenId);

      /// @dev This emits when the approved address for an NFT is changed or
      ///  reaffirmed. The zero address indicates there is no approved address.
      ///  When a Transfer event emits, this also indicates that the approved
      ///  address for that NFT (if any) is reset to none.
      event Approval(address indexed _owner, address indexed _approved, uint256
indexed _tokenId);

      /// @dev This emits when an operator is enabled or disabled for an owner.
      ///  The operator can manage all NFTs of the owner.
      event ApprovalForAll(address indexed _owner, address indexed _operator, bool
_approved);

      /// @notice Count all NFTs assigned to an owner
      /// @dev NFTs assigned to the zero address are considered invalid, and this
      ///  function throws for queries about the zero address.
      /// @param _owner An address for whom to query the balance
      /// @return The number of NFTs owned by `_owner`, possibly zero
      function balanceOf(address _owner) external view returns (uint256);

      /// @notice Find the owner of an NFT
      /// @dev NFTs assigned to zero address are considered invalid, and queries
      ///  about them do throw.
      /// @param _tokenId The identifier for an NFT
      /// @return The address of the owner of the NFT
      function ownerOf(uint256 _tokenId) external view returns (address);

      /// @notice Transfers the ownership of an NFT from one address to another
address
      /// @dev Throws unless `msg.sender` is the current owner, an authorized
      ///  operator, or the approved address for this NFT. Throws if `_from` is
      ///  not the current owner. Throws if `_to` is the zero address. Throws if
      ///  `_tokenId` is not a valid NFT. When transfer is complete, this function
      ///  checks if `_to` is a smart contract (code size > 0). If so, it calls
      ///  `onERC721Received` on `_to` and throws if the return value is not
      ///  `bytes4(keccak256("onERC721Received(address,address,uint256,bytes)"))`.
      /// @param _from The current owner of the NFT
      /// @param _to The new owner
      /// @param _tokenId The NFT to transfer
      /// @param data Additional data with no specified format, sent in call to `_to`
      function safeTransferFrom(address _from, address _to, uint256 _tokenId, bytes
data) external payable;
```

```
    /// @notice Transfers the ownership of an NFT from one address to another
address
    /// @dev This works identically to the other function with an extra data
parameter,
    ///  except this function just sets data to ""
    /// @param _from The current owner of the NFT
    /// @param _to The new owner
    /// @param _tokenId The NFT to transfer
    function safeTransferFrom(address _from, address _to, uint256 _tokenId)
external payable;

    /// @notice Transfer ownership of an NFT -- THE CALLER IS RESPONSIBLE
    ///  TO CONFIRM THAT `_to` IS CAPABLE OF RECEIVING NFTS OR ELSE
    ///  THEY MAY BE PERMANENTLY LOST
    /// @dev Throws unless `msg.sender` is the current owner, an authorized
    ///  operator, or the approved address for this NFT. Throws if `_from` is
    ///  not the current owner. Throws if `_to` is the zero address. Throws if
    ///  `_tokenId` is not a valid NFT.
    /// @param _from The current owner of the NFT
    /// @param _to The new owner
    /// @param _tokenId The NFT to transfer
    function transferFrom(address _from, address _to, uint256 _tokenId) external
payable;

    /// @notice Set or reaffirm the approved address for an NFT
    /// @dev The zero address indicates there is no approved address.
    /// @dev Throws unless `msg.sender` is the current NFT owner, or an authorized
    ///  operator of the current owner.
    /// @param _approved The new approved NFT controller
    /// @param _tokenId The NFT to approve
    function approve(address _approved, uint256 _tokenId) external payable;

    /// @notice Enable or disable approval for a third party ("operator") to manage
    ///  all of `msg.sender`'s assets.
    /// @dev Emits the ApprovalForAll event. The contract MUST allow
    ///  multiple operators per owner.
    /// @param _operator Address to add to the set of authorized operators.
    /// @param _approved True if the operator is approved, false to revoke
approval
    function setApprovalForAll(address _operator, bool _approved) external;
```

```
        /// @notice Get the approved address for a single NFT
        /// @dev Throws if `_tokenId` is not a valid NFT
        /// @param _tokenId The NFT to find the approved address for
        /// @return The approved address for this NFT, or the zero address if there is
none
        function getApproved(uint256 _tokenId) external view returns (address);

        /// @notice Query if an address is an authorized operator for another address
        /// @param _owner The address that owns the NFTs
        /// @param _operator The address that acts on behalf of the owner
        /// @return True if `_operator` is an approved operator for `_owner`, false
otherwise
        function isApprovedForAll(address _owner, address _operator) external view
returns (bool);
    }

    interface ERC165 {
        /// @notice Query if a contract implements an interface
        /// @param interfaceID The interface identifier, as specified in ERC-165
        /// @dev Interface identification is specified in ERC-165. This function
        ///  uses less than 30,000 gas.
        /// @return `true` if the contract implements `interfaceID` and
        ///  `interfaceID` is not 0xffffffff, `false` otherwise
        function supportsInterface(bytes4 interfaceID) external view returns (bool);
    }

    interface ERC721TokenReceiver {
        /// @notice Handle the receipt of an NFT
        /// @dev The ERC721 smart contract calls this function on the
        /// recipient after a `transfer`. This function MAY throw to revert and reject the
transfer. Return
        /// of other than the magic value MUST result in the transaction being reverted.
        /// @notice The contract address is always the message sender.
        /// @param _operator The address which called `safeTransferFrom` function
        /// @param _from The address which previously owned the token
        /// @param _tokenId The NFT identifier which is being transferred
        /// @param _data Additional data with no specified format
        /// @return
`bytes4(keccak256("onERC721Received(address,address,uint256,bytes)"))`
```

```
            /// unless throwing
            function onERC721Received(address _operator, address _from, uint256
_tokenId, bytes _data) external returns(bytes4);
        }
```

———

ERC-721 started as a EIP draft written by @dete and first came to life in the CryptoKitties project by Axiom Zen

ERC-721 has since moved out of beta and into a community formalized v1 spec, supported and endorsed by a large number of projects from across the crypto ecosystem. This EIP wouldn't be possible without time and dedication from a number of community members. Special thanks to @fulldecent and others who have put in tremendous amounts of work to push this forward.

ERC721 links:

http://erc721.org/
https://github.com/cryppadotta/dotta-license

https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/contracts/token/ERC721/ERC721BasicToken.sol

https://github.com/fulldecent/erc721-example

Sample marketplaces:

https://opensea.io/
https://rarebits.io/
https://www.emoon.io/

EIP 721:ERC-721
https://eips.ethereum.org/EIPS/eip-721

ERC-721 Structure

The ERC721 standard intends to serve as a standard API for non-fungible tokens within a smart contract.

The standard has the following functions and events in the smart contract which enable usage, ownership, and transfer of non fungible tokens (NFT's)

According to the specification, each NFT is identified by a unique 256-bit positive integer. The pair of (address, uint256) where uint256 is an unsigned integer is a unique identifier of a nonfungible asset on the blockchain. This integer is equivalent to a unique identifier that never changes and is bound to that specific asset.

The specification draws several functions from the ERC-20 specification to provide backward compatibility.

name – Name of the token/asset

symbol – Symbol of the token/asset

balanceOf – the balance of a given address

totalSupply – total supply of the asset/token

Apart from this, there are the standard transfer and ownership functions that are critical to the functioning of an NFT contract which we will not be covering in this article. For an in-depth insight into the specification, head over to read the details of EIP-721 on GitHub.

ERC-998

ERC998 is a proposal which aims to introduce a standard for Crypto Composables. It aims to act as a standard extension for any NFT to own other non-fungible ERC-721 tokens and/or fungible ERC-20 tokens. Transferring a single ERC998 token would mean that all the ERC-721 and/or ERC-20 tokens lying underneath would also change ownership. The author explains that having such a standard proposal would allow for a wide range of applications within ERC-721 tokens. For example, a cryptokitty could be a composable NFT token which includes unique accessories such as a pet bowl, clothing etc. And the bowl could hold fungible food tokens. Having a standard such as this would allow for seamless transfer of cryptoassets which include other cryptoassets embedded within themselves. This would not only increase the potential of cryptoassets on the blockchain and widen the possibilities of DApps such as Cryptokitties but also bring blockchain technology closer to mainstream adoption when it comes to real-world assets.

Overall, the ecosystem is highly underdeveloped when it comes to nonfungible assets, and we should see lots of activity in the space going forward. Representing nonfungible assets on the blockchain is a huge milestone, and the number of platforms and tools facilitating easier usage of NFT's are still unfolding, which implies the fact that we are scratching the surface of something big and venturing deeper into interesting spaces.

So why would anyone want a non-fungible token or smart contract if you cannot spend it?

These types of tokens aren't currencies. Rather, they represent unique assets that are represented on the Ethereum blockchain as smart contracts.

There was a pretty interesting event that sparked this proposal. You may remember a popular application called Cryptokitties that was released in December 2018. While transactions took place in cryptocurrency and data was imprinted onto the blockchain, the actual Cryptokitties did not exist on the chain.

Before then, there was nothing that could represent a unique asset. As a result, ERC-721 was introduced, and now these non-fungible tokens could represent assets like Cryptokitties.

What other assets could be represented as these non-fungible tokens?

Limiting the impact and application of ERC-721 to Cryptokitties would be a mistake. There are so many different potential use cases that ERC-721 opens up. For example, the tracking of ownership or a deed just became that much easier to accomplish on the Ethereum blockchain. Furthermore, any title, document, unique contract, or other asset is now able to be represent on the blockchain.

MOre detail on erc-721 Steemit:

As a relatively new standard, these types of tokens offer an important extension in Ethereum smart contracts, giving way to automating many aspects of a modern society.

What is a non-fungible ERC-721 token

The ERC-721 token standard was formed to address the need within the Ethereum core developer community to aid in creating applications for trading and tracking ownership of non-fungible digital or physical assets. Unlike in the ERC-20 token standard where individual tokens represent units of currency, the ERC-721 tokens are distinct from one another. The individuality of the ERC-721 tokens is explained by additional fields in the token standard;

The token has a specified name.

The token is given an individual token ID.

The contracts implemented in the token standard can track an arbitrarily large number of ERC-721 tokens, known as NFT's or non-fungible tokens, or deeds, simultaneously. This means that a single contract can contain one or more such tokens.

-Benefits of non-fungible tokens in labeling and sales of physical assets

Non-fungible tokens, or deeds, are ownership titles to property in the form of digital assets.
Title deeds open up a world of opportunities in digital contracts not seen before, with fungible, ERC-20 -type tokens. As ERC-721 -tokens represent user-defined real-world assets, the tokens can be used to mark, track and license ownership of these assets, and to label them by assigning the token's corresponding real-world item the token's

TokenID and reading the token's transaction hash. This is done via the ERC-821 standard, or Distinguishable Assets Registry (DAR for short):

Sample implementation:

https://github.com/decentraland/erc721/blob/master/ERC821.md

A reference implementation is produced for example by the Decentraland project:

https://github.com/decentraland/erc821

PRACTICAL APPLICATIONS OF NON-FUNGIBLE ERC-721 TOKENS:

Non-fungible tokens are in practical sense, title deeds to any property or certificate. The token doesn't define whether the property or certificate is digital or physical - or civil - in nature. It is up to the user to choose and define the practical use of the non-fungible token.

To successfully implement an ERC-721 -token into a business application, the user should consider some recurring issues.

-Smart contracts

For automating business contracts, the token should be governed with a smart contract that refers to and imports the token for practical purposes.

These contracts are often found in the following types of applications:

-Proof of origin
-Proof of ownership
-Proof of authenticity
-Licensing and royalties
-Leasing and renting, subscriptions
-Legal contracts
-Civil and public documents
-Notary documents on sales contracts

-KYC/AML
-Installments, obligations and loan contracts
-Auctions
-Supply chain management

The community may dream up many more possible use case scenarios.

-Labeling

Information contained within the token can be imported into a product label, which gives the user an instant access to the smart contract. Labeling can be used to scan the contract state, to verify the token's authenticity and to perform functions on the contract through an appropriate app that may import the contract.

Common ways to use labeling in practical sense include:

-RFID or NFC tags
-QR codes

-Sales of non-fungible ERC-721 tokens:

Non-fungible tokens require a wallet that supports them, ie. Metamask at the time of writing this document. A good rule of thumb is that what supports Cryptokitties, supports every non-fungible token.

Sales of non-fungible tokens on an exchange is closer to an application of a marketplace than speculative asset exchange.

-Decentralized asset exchanges

Decentralized asset exchanges may benefit from non-fungible tokens, by allowing peer-to-peer transfer of title deeds. This is especially useful in games or in civil contracts, or in simple business contracts that do not require a notary or a public third party for compliance or verification.

There are proposed atomic swaps for non-fungible tokens as follows:

https://github.com/ethereum/EIPs/issues/875

-Internal asset exchanges

An e-commerce application may be built to support non-fungible tokens as certificates of products offered by the seller within an internal marketplace. This kind of closed-loop merchant-customer asset exchange is relevant especially in e-commerce, real estate development, rental, sharing economy and utilities industries, and also in fine art and media industries. Internal asset exchanges provide a practical way to distribute valid certificates of ownership and proofs of authenticity.

-Art gallery example of non-fungible ERC-721 tokens in a practical business application.

An art gallery hosts a marketplace for paintings. The platform of the art gallery transfers ownership of the paintings to the customer, governed by a licensing contract set by the artist. Each painting is represented by a non-fungible token, encoded within the artwork itself.

The non-fungible token can track:

-The origin of the work, or who painted it.
-The current owner.
-The licensing contract and whether the conditions are met.
-Relevant information about the artwork.
-The sales contract(s) associated with the token.

The non-fungible token and the sales contract(s) associated with it provide the artist means of copyright and licensing in the digital world, and fine artists means of distinguishing their original artwork from forgery.

The non-fungible token provide the gallerist means to prove that their hosted artwork is indeed authentic, as well as means of offering custom contracts such as renting, licensing, exhibitions and direct sales.
The non-fungible token provides the customer proof of authenticity and flexible deals on fine art both in collection and commerce.

-Public asset exchanges

Public asset exchanges benefit from non-fungible tokens especially in e-commerce and related activities. Non-fungible tokens can incorporate data of the attributes and supply chain of the product, which makes product certification and enterprise resource planning more streamlined, through a trustless blockchain -enabled platform.
For example, a food product or an industrial may be tracked all the way to the origin and processes and ingredients, and middlemen may be identified and tracked through a contract associated with a non-fungible token representing a product.

Media distributors benefit from non-fungible tokens by offering licenses associated with the token directly on a public platform.

Sharing economy operators, real estate developers and rental agencies may distribute contracts associated with their fleet of assets directly on their public platform, for example within a mobile app.

Possible business applications of non-fungible ERC-721 tokens:

-Fine art
-Film and music royalties
-Online gaming
-Diamond and gemstone market
-Ticketing
-Physical property
-Loans and obligations
-Digital goods
-Distributed orders
-E-commerce
-Certified goods
-Civil certificates
-Decentralized ERP

What is required for the applications to pan out in real life?

-App development

App developers may integrate ERC-721 tokens in their platforms, though to streamline this development, public API's and public network nodes could make non-fungible blockchain tokens accessible to the bulk of mobile and web developers - most of whom use JavaScript.

-Pilot projects

Successful pilot projects have been proven to inspire further development within the blockchain developer community, namely through successful startups, and socially aware projects such as ConsenSys. Pioneers often reap rewards, which encourage other players to enter a new, competitive market.

-More 'kitties

Cryptokitties was the first successful implementation of a non-fungible token standard into a real world application. The game took the Internet by storm, and at one point represented the bulk of Ethereum transactions. Hit products such as Cryptokitties can bring the technology into the public view probably more effectively than the more boring methods. However, to make sure the public will adopt the application, it should be intuitive to use.
I cannot stress enough how important it is to have an accessible mobile and web wallet. The wallet is the first experience the user encounters, therefore the wallet should be as non-technical as possible for widepread adoption.

Conclusion

Non-fungible tokens may transform our society in ways never seen before since the invention of the printing press, by taking out the middleman in business contracts and bureaucracy.

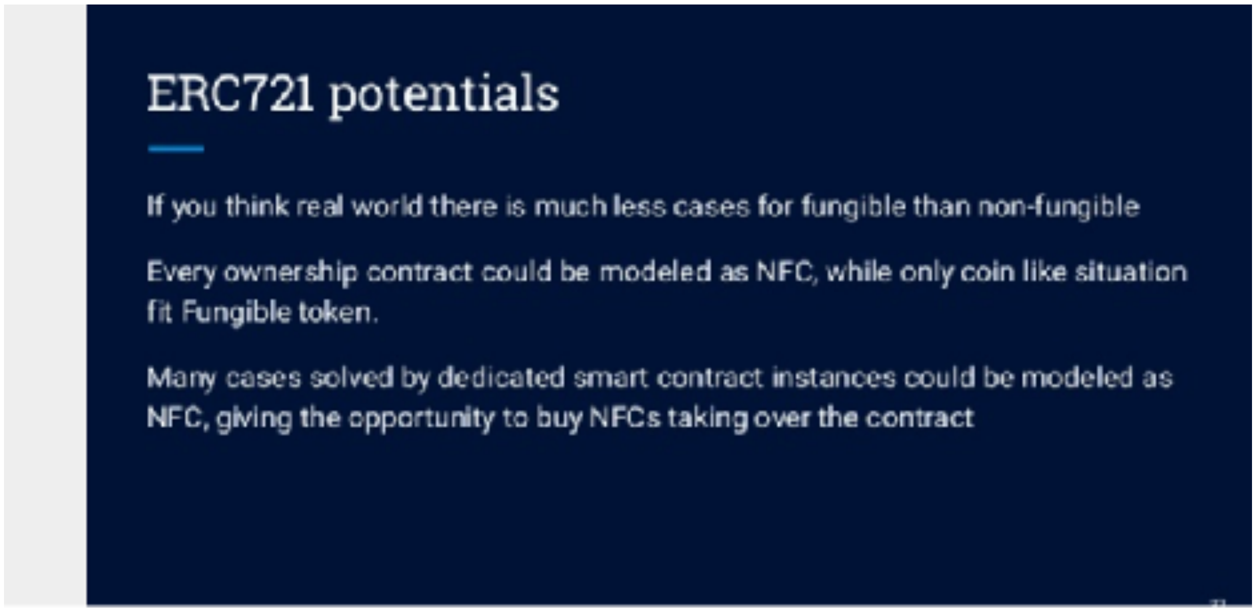
Many if not most tasks today delegated to public servants, accountants, secretaries, commercial agents and opaque middlemen can be automated with a transparent, trustless system. This new method will liberate us from boring tasks and uncertainty in many aspects of modern life.

The transition will be rapid, once the foundation is laid.

Look at the 998

Erc998- composable heierarchy
https://medium.com/coinmonks/introducing-crypto-composables-ee5701fde217

https://medium.com/coinmonks/crypto-composables-erc-998-update-1cc437c13664 https://github.com/ethereum/EIPs/issues/998#issuecomment-381563361

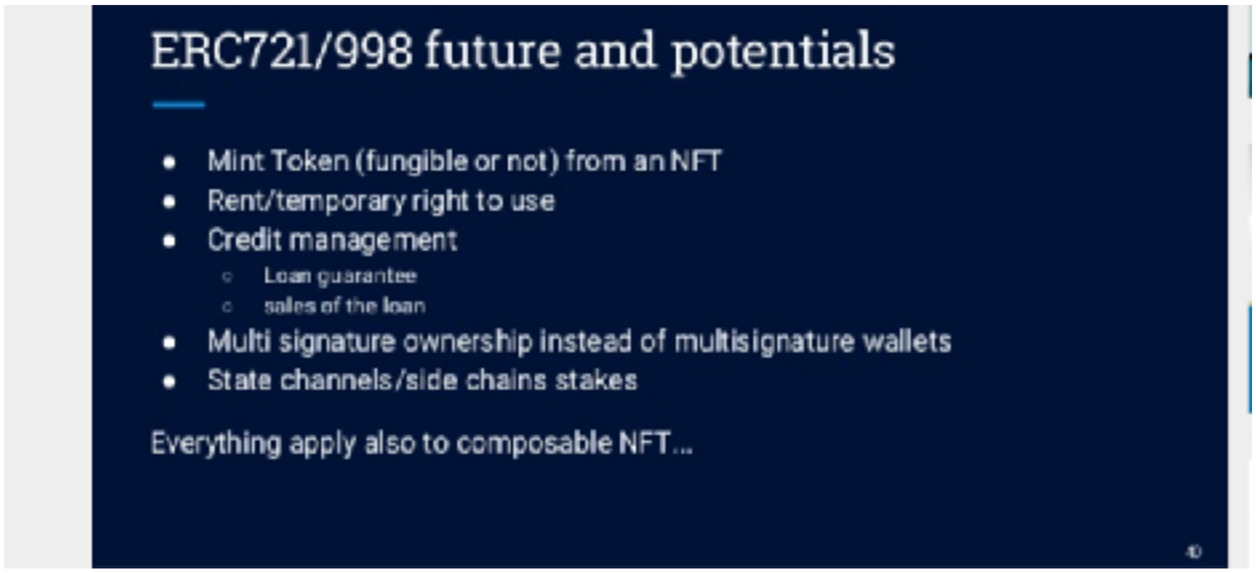


ERC721 potentials

If you think real world there is much less cases for fungible than non-fungible

Every ownership contract could be modeled as NFC, while only coin like situation fit Fungible token.

Many cases solved by dedicated smart contract instances could be modeled as NFC, giving the opportunity to buy NFCs taking over the contract



ERC721/998 future and potentials

- Mint Token (fungible or not) from an NFT
- Rent/temporary right to use
- Credit management
  - Loan guarantee
  - sales of the loan
- Multi signature ownership instead of multisignature wallets
- State channels/side chains stakes

Everything apply also to composable NFT...