NFT Making on the IC

MY NOTES AS A BEGINNER IN SUPPORT OF MOTOKO BOOTCAMP

@tedreinhardt

Ver 0.9
22 Feb 2022

Aim

- Share what I have been learning about creating NFTs so that others may be able to advance their own projects.

- I have very little experience. I minted my first 360 degree NFT in August/2021 using DepartureLabs Minter. Here is the NFT content.
  - https://jb6ng-naaaa-aaaaf-qacvq-cai.raw.ic0.app/mmiwg.html

Disclaimer

- These are my notes and may contain inaccuracies.
- Use the information at your own risk.

A word of advice: Never share your wallet recovery seed with anyone.

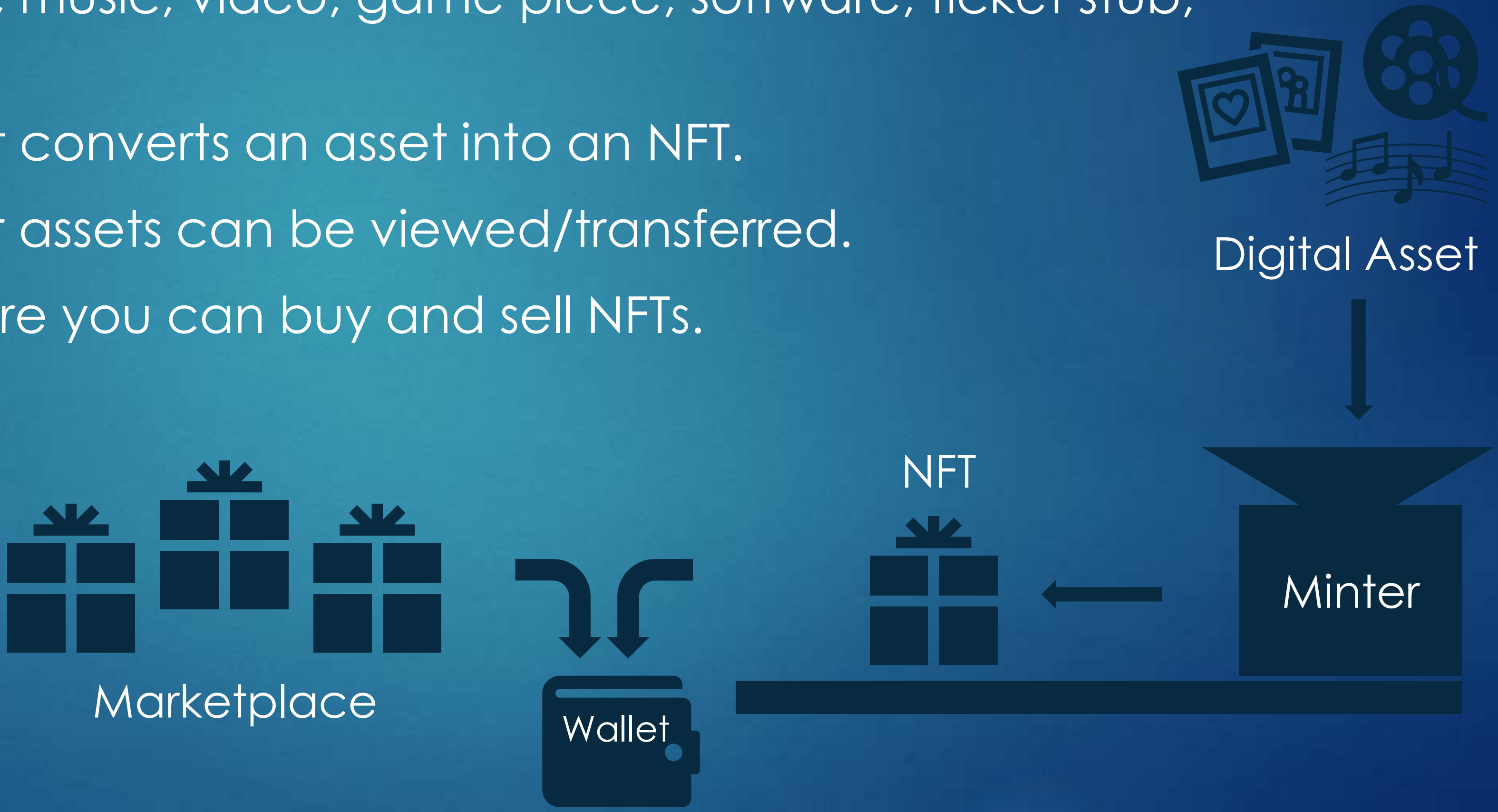
Non-Fungible Token (NFT)

- A non-fungible token (NFT) is a unique and non-interchangeable unit of data stored on a digital ledger (blockchain).
- NFTs can be associated with reproducible digital files such as photos, videos and audio.
- NFTs use a digital ledger to provide a public certificate of authenticity or proof of ownership, but it does not restrict the sharing or copying of the underlying digital file.
- The lack of interchangeability (fungibility) distinguishes NFTs from blockchain fungible token such as ICP and Bitcoin.

Source: Wikipedia contributors. (2021, December 15). Non-fungible token.
In Wikipedia, The Free Encyclopedia. Retrieved 19:53, December 15, 2021, from https://en.wikipedia.org/w/index.php?title=Non-fungible_token&oldid=1060457068

The NFT terminology

- Digital asset: photo, music, video, game piece, software, ticket stub, etc.
- Minter: process that converts an asset into an NFT.
- Wallet – where your assets can be viewed/transferred.
- Marketplace – where you can buy and sell NFTs.



Difference between a digital asset and a NFT?

Digital Asset + Certificate of ownership +

Built-in transfer, registration and access services.

Optionally a program to transform or display the asset or deliver a service.

An NFT

The IC enables you to do all this on chain in a canister (smart contract).
It is also economical on the IC.

Costs of minting an NFT on the IC

- NFTs are stored in canisters.
- Minting a canister costs about $3 USD on the IC:
  - Canister creation costs 2 trillion Cycles.
  - 1 trillion Cycles costs 1 Standard Define Rate (SDR), a term defined by the International Monetary Fund or approximately $1.39 USD as at 22 Feb 2022.
- In addition, there is an ongoing fee for storage, transmittal and processing (very low paid in Cycles).
- Over time, the cost of storage and execution will have to be topped up unless it is loaded with more Cycles to maintain it. If not, it will be removed from the blockchain when it is depleted.

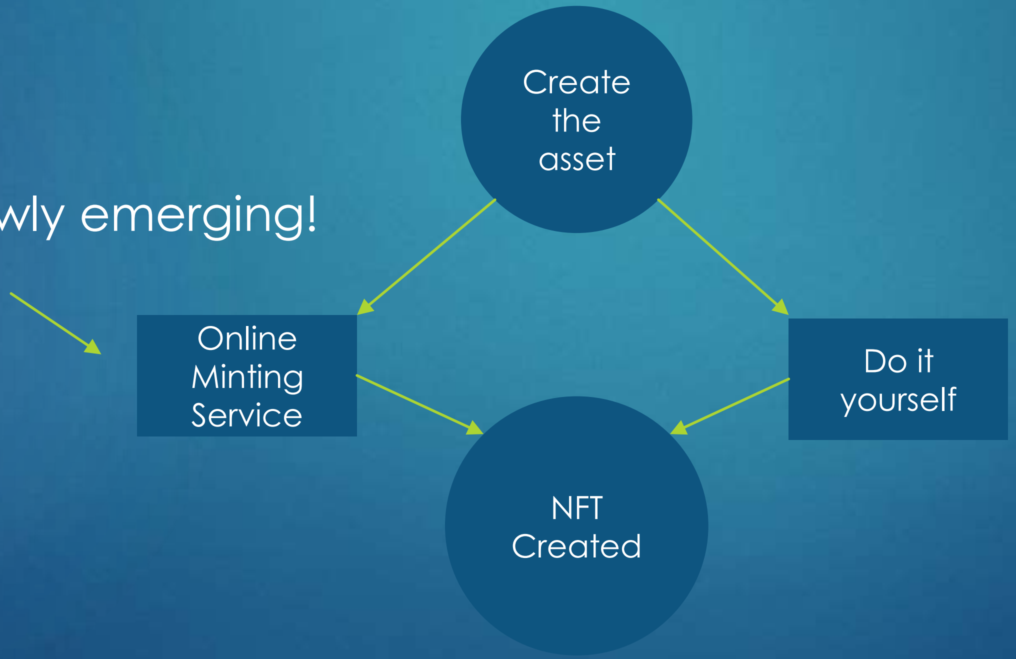
Multiple NFTs can also be stored in a single canister reducing cost.

- Using a multi-NFT canister, minting 100 000 tokens would cost around $3 vs $300 000.

How do you make an NFT?

- Create a digital asset , e.g. an image saved as a .png file. (Any digital asset supported by the minter/marketplace can be used.)
- Use an online minter to create the NFT or do it yourself.

They are slowly emerging!



Online minting of your content

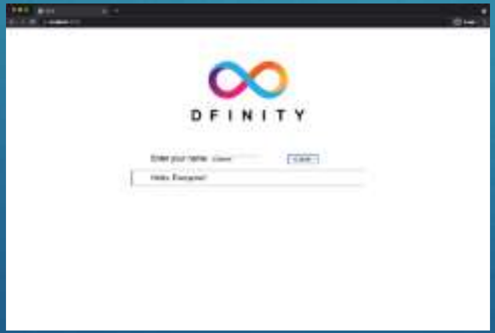
- DepartureLabs had an experimental minter – offline now.
- NFT anvil is in development.

  https://5rttq-yqaaa-aaaai-qa2ea-cai.raw.ic0.app/mint
- Toniqlabs will mint collections as part of the collect intake process if deemed of interest to Entrepot.
- BobNFTs.com is a new minting service with potential.
- NFTStudio.biz is a new minting service with potential.
- ICME are working on a minting offering.
- Blocks Editor has a DIP721 example which can be built to mint and learn.

Let me know if you find any others.

Create it yourself

- You will need to have the DFINITY Canister SDK and dependencies installed and running on a computer using :
  - Windows with WSL2 and Linux subsystem installed (e.g. Ubuntu2004);
  - Linux; or
  - Mac
- Follow the quick start tutorial and deploy the sdk Hello example.
- Test to see if you can see the front end that looks like this:

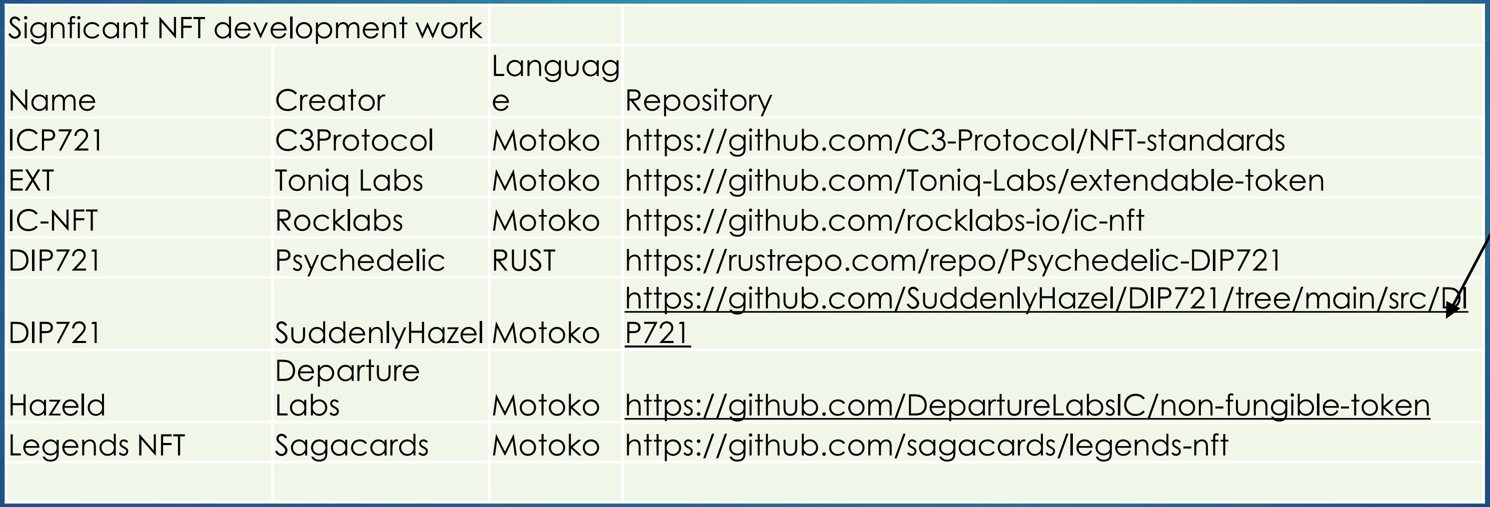


So what did that achieve?

- Following the quick start tutorial means you created a smart contract owned by you that displays an image (the dfinity symbol) stored on the local replica. It has an index.html, index.js and the dfinity logo png.
- Turning into an NFT means adding:
  - the NFT data elements for the certificate of ownership data
  - the NFT smart contract functions to enable minting, transfer, etc
  - digital assets.

ERC-721

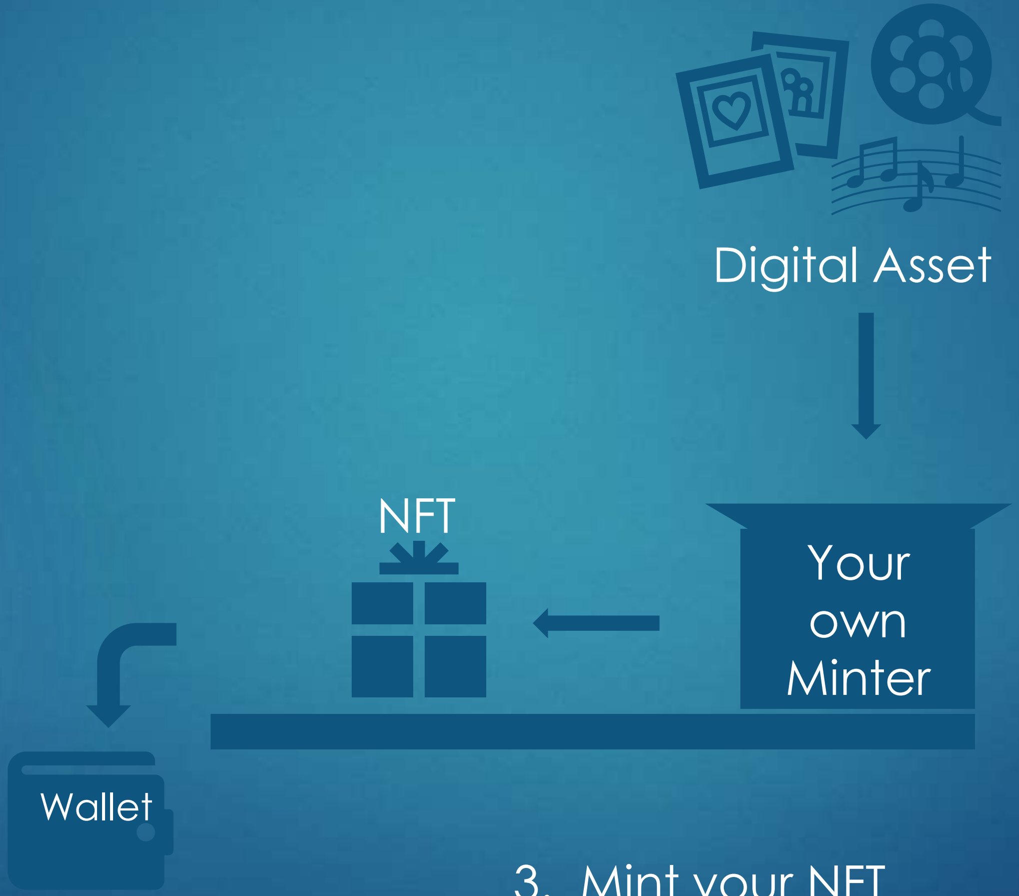
- A standard called ERC-721 (ERC is Ethereum Request for Comment) defines data elements and methods that must be present when constructing an NFT for trading within the Ethereum eco-system to aid interoperability with wallets and ecosystems.
- The Internet Computer community has not defined a standard. Instead a variety of implementations are in use that define data elements and functions.



Signficant NFT development work			
Name	Creator	Languag e	Repository
ICP721	C3Protocol	Motoko	https://github.com/C3-Protocol/NFT-standards
EXT	Toniq Labs	Motoko	https://github.com/Toniq-Labs/extendable-token
IC-NFT	Rocklabs	Motoko	https://github.com/rocklabs-io/ic-nft
DIP721	Psychedelic	RUST	https://rustrepo.com/repo/Psychedelic-DIP721
DIP721	SuddenlyHazel	Motoko	https://github.com/SuddenlyHazel/DIP721/tree/main/src/DIP721
Hazeld	Departure Labs	Motoko	https://github.com/DepartureLabsIC/non-fungible-token
Legends NFT	Sagacards	Motoko	https://github.com/sagacards/legends-nft

We use one derived from DIP721.

The workflow



1. Create the asset
2. Build your minter
3. Mint your NFT
4. Load it in your wallet

Minting with a @BlocksEditor DIP721 example

DIP721 Example

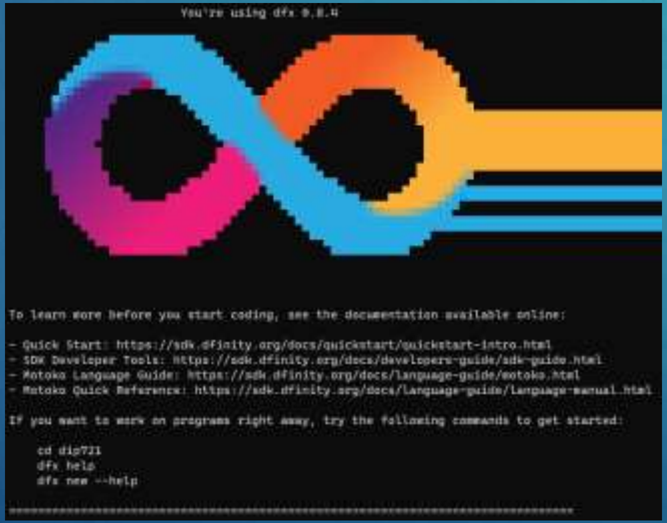
- DIP721 “standard” was developed in Rust by @PsychedelicDAO
- @SuddenlyHazel implemented DIP721 in Motoko.
- @BlocksEditor adapted the @SuddenlyHazel’s version and will be used as the example. It is found here: https://blocks-editor.github.io/
- It differs from the DIP721 implementation in that the functions are not using the DIP721 postfix namespace and only a subset of DIP721 functions are implemented.

How to build (after you installed the SDK)

- Create a new project called dip721

```
$ dfx new dip721
```

- It will install and when successful displays the following.



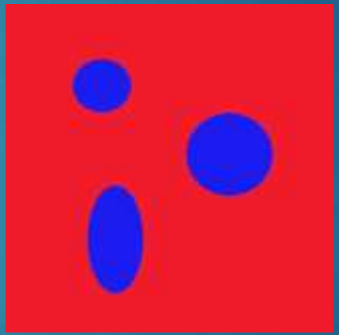
```
                   You're using dfx 0.8.4

To learn more before you start coding, see the documentation available online:

- Quick Start: https://sdk.dfinity.org/docs/quickstart/quickstart-intro.html
- SDK Developer Tools: https://sdk.dfinity.org/docs/developers-guide/sdk-guide.html
- Motoko Language Guide: https://sdk.dfinity.org/docs/language-guide/motoko.html
- Motoko Quick Reference: https://sdk.dfinity.org/docs/language-guide/language-manual.html

If you want to work on programs right away, try the following commands to get started:

    cd dip721
    dfx help
    dfx new --help

==============================================================================
```

DIP721 - Create your digital assets

- Create 3 digital assets and store them in into the ~/dip721/src/dip721_assets/assets subdirectory naming them 1.png, 2.png and 3.png.

1.png

2.png


3.png

Deploy the canister locally and test

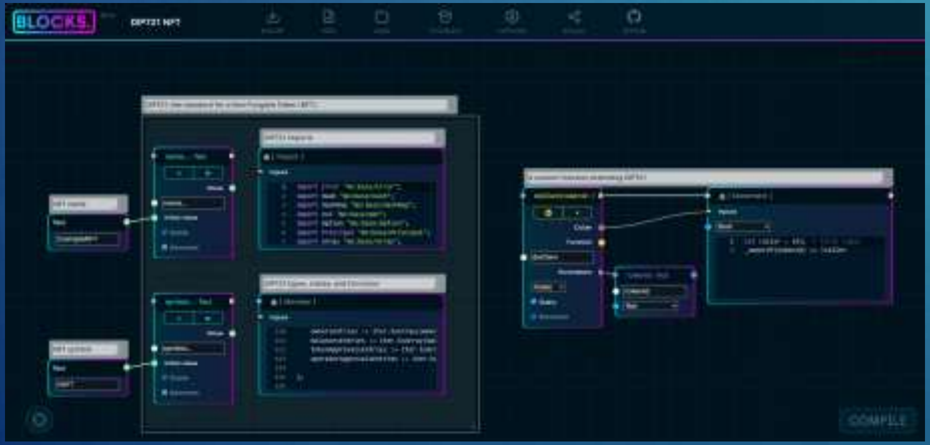
While in the top project directory:

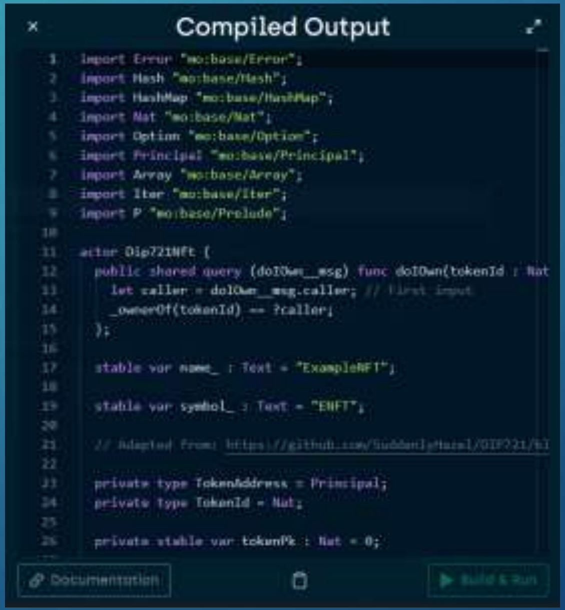
- dfx start (in a separate window as it starts the replica processes)
- dfx deploy
- npm start
- In a browser go to: http://localhost:8080/1.png

Your first image should be displayed.

Add the DIP 721 smart contract code

- Go to https://blocks-editor.github.io/
- Load the DIP721 example and compile
- Copy the compiled output (the source code) and paste the contents of main.mo into ~/dip721/src/dip721/main.mo





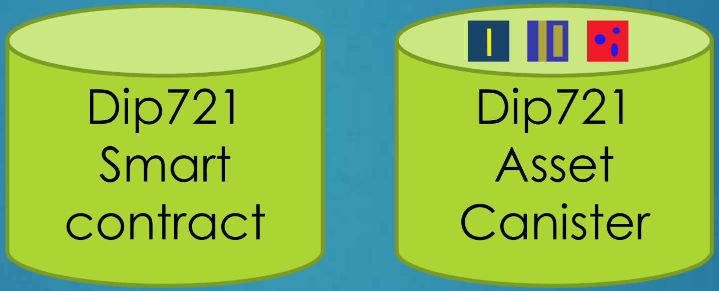
```
import Error "mo:base/Error";
import Hash "mo:base/Hash";
import HashMap "mo:base/HashMap";
import Nat "mo:base/Nat";
import Option "mo:base/Option";
import Principal "mo:base/Principal";
import Array "mo:base/Array";
import Iter "mo:base/Iter";
import P "mo:base/Prelude";

actor Dip721NFt {
  public shared query (doIOwn__msg) func doIOwn(tokenId : Nat
    let caller = doIOwn__msg.caller; // First input
    _ownerOf(tokenId) == ?caller;
  };

  stable var name_ : Text = "ExampleNFT";

  stable var symbol_ : Text = "ENFT";

  // Adapted from: https://github.com/SuddenlyHazel/DIP721/bl

  private type TokenAddress = Principal;
  private type TokenId = Nat;

  private stable var tokenPk : Nat = 0;
```

main.mo

Deploy the smart contract

- dfx deploy
- npm start

Results in 2 canisters



Let's try it out.

Show the name of the NFT

```
$ dfx canister call dip721 name
("ExampleNFT")
```

Show the name of the Symbol

```
$dfx canister call dip721 symbol
("ENFT")
```

This is the Uniform Resource Identifier where the asset is stored

Mint a token using the first 1.png as the digital asset

```
$dfx canister call dip721 mint http://localhost:8080/1.png
(1 : nat)
```

This is the TokenId

Show the owner of TokenID 1.

```
$dfx canister call dip721 ownerOf '(1)'
(
  opt principal "p3oiq-zvq7o-ir4je-ngxhk-br4ps-ymn3e-i7lsc-a6o67-irbt3-h5ddq-pae",
)
```

Transfer the token to someone else.

TokenID 1

Confirm you own it.

```
$dfx canister call dip721 doIOwn '(1)'
(true)
```

Transfer it.

```
$dfx canister call dip721 transferFrom '(principal "p3oiq-zvq7o-ir4je-ngxhk-br4ps-ymn3e-i7lsc-a6o67-irbt3-h5ddq-pae",principal "zht7g-jivec-azc2g-f5bkj-oxsfc-nvyo6-ch4jb-el2tb-ebyuf-zeo4d-gae",1)'
(NULL)
```

Confirm you don't own it.

```
$dfx canister call dip721 doIOwn '(1)'
(false)
```

Who is the owner now?

```
$dfx canister call dip721 ownerOf '(1)'

(
  opt principal "zht7g-jivec-azc2g-f5bkj-oxsfc-nvyo6-ch4jb-el2tb-ebyuf-zeo4d-gae",
)
```

@BlocksEditor DIP721 has more methods

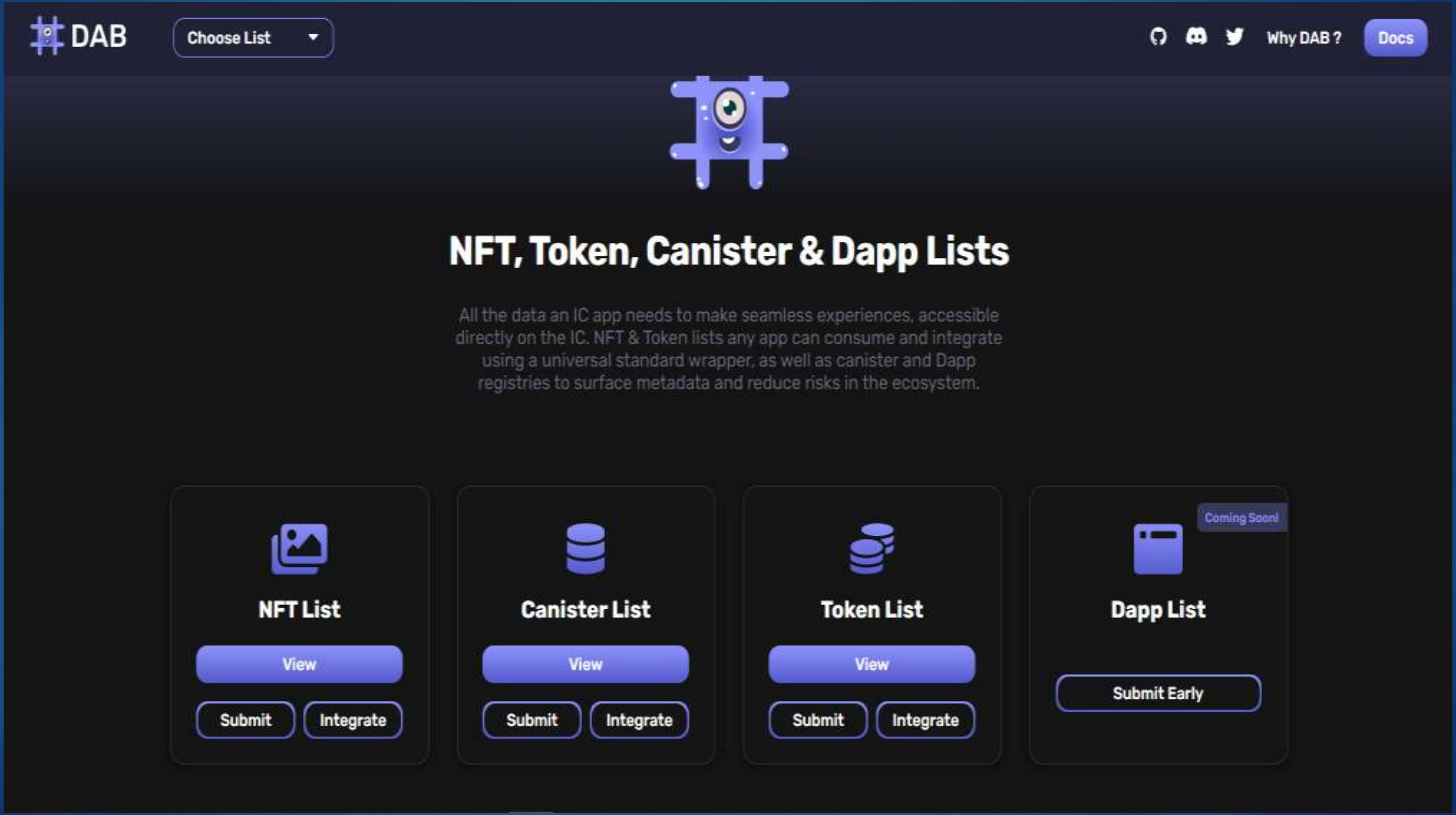
- approve(to : Principal, tokenId : TokenId)
- balanceOf(p : Principal)
- doIOwn(tokenId : Nat)
- getApproved(tokenId : Nat)
- isApprovedForAll(owner : Principal, opperator : Principal)
- mint(uri : Text)
- name()
- ownerOf(tokenId : TokenId)
- setApprovalForAll(op : Principal, isApproved : Bool)
- symbol()
- tokenURI(tokenId : TokenId)
- transferFrom(from : Principal, to : Principal, tokenId : Nat)

This is not prod ready yet.

- No front end.
- Marketplace integration.
- Lacking wallet integration
- Access control and security
- The example does not implement the full DIP721 interface standard. DFINITY is publishing this in the coming weeks as part of their tutorial.
- Ownership history is missing
- Integration with other Wallets and Marketplaces via DAB SDK has not been implemented.

Dip721 Smart contract
Dip721 Asset Canister

DAB



NFT, Token, Canister & Dapp Lists

All the data an IC app needs to make seamless experiences, accessible directly on the IC. NFT & Token lists any app can consume and integrate using a universal standard wrapper, as well as canister and Dapp registries to surface metadata and reduce risks in the ecosystem.

CAP

- Certified Asset Provenance (CAP)

- CAP is an open internet service providing transaction history & asset provenance for NFT's & Tokens on the Internet Computer. It solves the huge IC problem that assets don't have native transaction history, and does so in a scalable, trustless and seamless way so any NFT/Token can integrate with one line of code.

Clean up the old code

- Get rid of the leftover assets, html, css and etc. from the greet application create by the dfx new process and re-deploy.

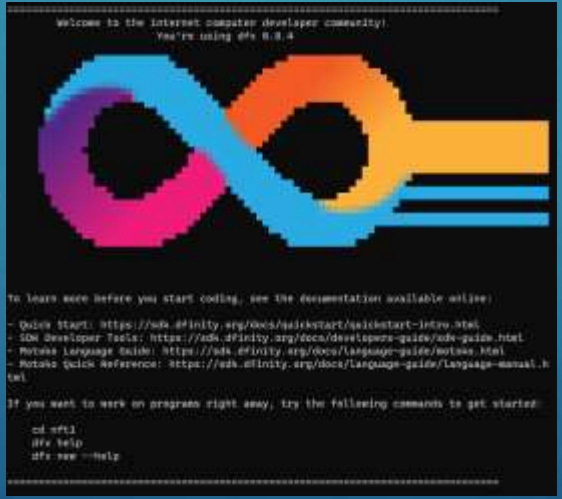
EXT Standard

Steps to build (after you installed the SDK)

- Create a new project called nft1

```
$ dfx new nft1
```

- It will install and, when successful, display the following:



```
===============================================================================
        Welcome to the internet computer developer community!
                    You're using dfx 0.8.4

To learn more before you start coding, see the documentation available online:

- Quick Start: https://sdk.dfinity.org/docs/quickstart/quickstart-intro.html
- SDK Developer Tools: https://sdk.dfinity.org/docs/developers-guide/sdk-guide.html
- Motoko Language Guide: https://sdk.dfinity.org/docs/language-guide/motoko.html
- Motoko Quick Reference: https://sdk.dfinity.org/docs/language-guide/language-manual.h
tml

If you want to work on programs right away, try the following commands to get started:

    cd nft1
    dfx help
    dfx new --help

===============================================================================
```

Clone the NFT code repository from GITHUB

- Change directory into ~nft1.
- Install github (if not already done google for your particular platform).
- Clone the repository (repo) locally.

```
$ git clone git@github.com:Toniq-Labs/extendable-token.git
```

- Copy the erc721.mo file in the examples subdirectory into the ~nft1/src/nft1 subdirectory.
- Rename the existing main.mo to main.old or delete it.
- Rename the erc721.mo to main.mo

This replaces the Motoko code with the ERC721 code.

Move the dependencies

- Copy the extendable-token motoko sub-directory and its contents into the ~nft1/src directory.
- Check ext and util directories exist and the files are there:

```
$ ls -lR motoko
motoko:
total 8
drwxr-xr-x 2 jhm jhm 4096 Dec 16 10:24 ext
drwxr-xr-x 2 jhm jhm 4096 Dec 16 10:24 util

motoko/ext:
total 48
-rw-r--r-- 1 jhm jhm  621 Dec 16 10:24 Allowance.mo
-rw-r--r-- 1 jhm jhm  831 Dec 16 10:24 Archive.mo
-rw-r--r-- 1 jhm jhm  445 Dec 16 10:24 Batch.mo
-rw-r--r-- 1 jhm jhm  550 Dec 16 10:24 Common.mo
-rw-r--r-- 1 jhm jhm 6672 Dec 16 10:24 Core.mo
-rw-r--r-- 1 jhm jhm  400 Dec 16 10:24 Fee.mo
-rw-r--r-- 1 jhm jhm  701 Dec 16 10:24 Ledger.mo
-rw-r--r-- 1 jhm jhm  394 Dec 16 10:24 NonFungible.mo
-rw-r--r-- 1 jhm jhm  999 Dec 16 10:24 Operator.mo
-rw-r--r-- 1 jhm jhm  546 Dec 16 10:24 Secure.mo
-rw-r--r-- 1 jhm jhm  202 Dec 16 10:24 Subscribe.mo

motoko/util:
total 20
-rw-r--r-- 1 jhm jhm 1564 Dec 16 10:24 AccountIdentifier.mo
-rw-r--r-- 1 jhm jhm 3991 Dec 16 10:24 CRC32.mo
-rw-r--r-- 1 jhm jhm 1866 Dec 16 10:24 Hex.mo
-rw-r--r-- 1 jhm jhm 5392 Dec 16 10:24 SHA224.mo
```

Start the local replica

In a separate window/terminal session.

- dfx start --clean   warning: --clean wipes out the old replica state

OR

- dfx start

```
$ Dfx start --clean
Starting webserver for /_/
binding to: 127.0.0.1:45621
Dec 18 11:30:16.222 INFO ic-starter. Configuration: ValidatedConfig { replica_path: Some("/home/ted/.cache/dfinity/versions/0.8.4/replica"),
replica_version: "0.8.0", log_level: Warning, cargo_bin: "cargo", cargo_opts: "", state_dir: "/home/ted/nft1/.dfx/state/replicated_state",
http_listen_addr: 127.0.0.1:0, http_port_file: Some("/home/ted/nft1/.dfx/replica-configuration/replica-1.port"), metrics_addr: None,
provisional_whitelist: Some(All), artifact_pool_dir: "/home/ted/nft1/.dfx/state/replicated_state/node-100/ic_consensus_pool", crypto_root:
"/home/ted/nft1/.dfx/state/replicated_state/node-100/crypto", state_manager_root: "/home/ted/nft1/.dfx/state/replicated_state/node-
100/state", registry_local_store_path: "/home/ted/nft1/.dfx/state/replicated_state/ic_registry_local_store", unit_delay: None, initial_notary_delay:
Some(600ms), detect_consensus_starvation: None, consensus_pool_backend: Some("rocksdb"), state_dir_holder: None }, Application: starter
Dec 18 11:30:16.223 INFO Initialize replica configuration "/home/ted/nft1/.dfx/state/replicated_state/ic.json5", Application: starter
Dec 18 11:30:17.154 INFO Executing "/home/ted/.cache/dfinity/versions/0.8.4/replica" "--replica-version" "0.8.0" "--config-file"
"/home/ted/nft1/.dfx/state/replicated_state/ic.json5", Application: starter
Dec 18 11:30:23.398 WARN s:knvjx-zgkm4-hkvvw-3ueag-xlh7y-zje4m-54tyb-vp2bf-xigrc-mrffa-xae/n:6ufqp-7ok73-oxrj6-zgchl-xllg5-dukck-4zeln-
dlz3g-megws-lov4f-qae/ic_p2p/download_management PeerManagerImpl::new(): relay_config = None
version: 0.7.0
 Dec 18 06:30:23.433 INFO Log Level: INFO
 Dec 18 06:30:23.433 INFO Starting server. Listening on http://127.0.0.1:8000/
```

Create an empty Canister

```
$ dfx canister create nft1
Creating a wallet canister on the local network.
The wallet canister on the "local" network for user "default" is "rwlgt-iiaaa-aaaaa-aaaaa-cai"
Creating canister "nft1"...
"nft1" canister created with canister id: "rrkah-fqaaa-aaaaa-aaaaq-cai"
```

Build the wasm code

- dfx build nft1

These warnings are not an issue

```
$ dfx build nft1
Building canisters...
/home/ted/nft1/src/motoko/util/Hex.mo:48.17-48.30: warning [M0154], field unwrap is deprecated:
Option.unwrap is unsafe and fails if the argument is null; it will be removed soon; use a `switch` or `do?` expression instead
/home/ted/nft1/src/motoko/util/Hex.mo:49.17-49.30: warning [M0154], field unwrap is deprecated:
Option.unwrap is unsafe and fails if the argument is null; it will be removed soon; use a `switch` or `do?` expression instead
/home/ted/nft1/src/motoko/util/Hex.mo:48.17-48.30: warning [M0154], field unwrap is deprecated:
Option.unwrap is unsafe and fails if the argument is null; it will be removed soon; use a `switch` or `do?` expression instead
/home/ted/nft1/src/motoko/util/Hex.mo:49.17-49.30: warning [M0154], field unwrap is deprecated:
Option.unwrap is unsafe and fails if the argument is null; it will be removed soon; use a `switch` or `do?` expression instead
/home/ted/nft1/src/motoko/util/AccountIdentifier.mo:38.14-38.27: warning [M0154], field unwrap is deprecated:
Option.unwrap is unsafe and fails if the argument is null; it will be removed soon; use a `switch` or `do?` expression instead
```

Get the principal

- dfx identity get-principal will show you the identity you are using on the local replica. Your principal response will be different to this.

```
$ dfx identity get-principal
zht7g-jivec-azc2g-f5bkj-oxsfc-nvyo6-ch4jb-el2tb-ebyuf-zeo4d-gae
```

Copy your result to the clipboard

Install the code into the canister

- The dfx canister install is used to move the code into the canister and to set the owner of the registry to your principal. Your principle will be different (the output from the dfx identity get-principal)

```
dfx canister install nft1 --argument="(principal \"zht7g-jivec-azc2g-f5bkj-oxsfc-nvyo6-ch4jb-el2tb-ebyuf-zeo4d-gae\")"
```

Change to match your principal from the previous step

Status check

- You now have a canister with the dfinity logo in it and the NFT logic to manage the token. It also has an HTML page to render it and a no-longer-needed index.js which can be removed.
- Check to see which identity will be the minter.

```
$ dfx canister call nft1 getMinter
(principal "zht7g-jivec-azc2g-f5bkj-oxsfc-nvyo6-ch4jb-el2tb-ebyuf-zeo4d-gae")
```

Should match your principal

Mint your first NFT

One line

- Call the mintNFT method passing the principal (vs a stoic address) with the appropriate principal

```
$dfx canister call nft1 mintNFT "(record { to = (variant { \"principal\" = principal \"zht7g-jivec-azc2g-f5bkj-oxsfc-nvyo6-ch4jb-el2tb-ebyuf-zeo4d-gae\" })} )“

(0 : nat32)
```

Result

Minted a NFT in position 0 of the registry.

Run it again and the next one will be in position 1

Sets the NFT token owner and may be different for each token.

You can use dfx to call public methods of the canister

- getMinter()   e.g. dfx canister call nft1 getMinter
- setMinter(minter : Principal)
- mintNFT(request : MintRequest)
- transfer(request: TransferRequest)
- approve(request: ApproveRequest)
- extensions()
- balance(request : BalanceRequest)
- allowance(request : AllowanceRequest)

...continued

- bearer(token : TokenIdentifier)
- supply(token : TokenIdentifier)
- getRegistry()
- getAllowances()
- getTokens()
- metadata(token
- public func acceptCycles()
- availableCycles()

This is what was achieved

- You minted an NFT with ownership information stored within the canister on the local replica.
- There is no access control implemented – anyone can see the content of the canister or call the methods – not just the NFT holder.
- Upgrading the contract may clobber persistent data (I have not checked this) so be aware.
- You can mint additional tokens by running the mint command again and it will store it in the same canister with an index increment.

Deploying an NFT costs Cycles

- Creating a canister costs 2 Trillion Cycles.

- Sign up to GITHUB to get free Cycles from the Dfinity Cycles faucet:

  https://faucet.dfinity.org/auth

- With a GITHUB account created more than 90 days ago, you can get free Cycles to test deployment on the IC.
- The Faucet will give about $20 worth of Cycles once.

Now moving to the real world

- Doing it on the IC involves:
  - creating a canister (incurs a cost of 2 x SDR) using the NNS or using dfx
  - Noting the canister ID.
  - Replacing the Dfinity logo image file with your digital asset image.
  - Cleaning up unused code (index.js and index.html).
  - **Adding access controls in the Motoko code.**
  - Installing the code into the canister on the IC with the correct principle.
  - Then running the dfx minter command pointing to the canister on the IC.

```
dfx deploy --network ic --argument="(principal \"zht7g-jivec-azc2g-f5bkj-oxsfc-nvyo6-ch4jb-el2tb-ebyuf-zeo4d-gae\")“
dfx canister --network ic call nft1 mintNFT "(record { to = (variant { \"principal\" = principal \"zht7g-jivec-azc2g-f5bkj-oxsfc-nvyo6-ch4jb-el2tb-ebyuf-zeo4d-gae\" }); metadata=opt vec
{92;120;48;65;116;105;100;34;114;121;106;108;51;45;116;121;97;97;97;45;97;97;97;97;97;45;97;97;97;98;97;45;99;97;105;48;48;48;49}  } )"
dfx canister --network ic call nft1 getTokens
```

Now moving to the real world

- Doing it on the IC involves:

```
dfx deploy --network ic --argument="(principal \"zht7g-jivec-azc2g-f5bkj-oxsfc-nvyo6-ch4jb-el2tb-ebyuf-zeo4d-gae\")"


dfx canister --network ic call nft1 mintNFT "(record { to = (variant { \"principal\" = principal \"zht7g-jivec-azc2g-f5bkj-oxsfc-nvyo6-ch4jb-el2tb-
ebyuf-zeo4d-gae\" }); metadata=opt vec
{92;120;48;65;116;105;100;34;114;121;106;108;51;45;116;121;97;97;97;45;97;97;97;97;97;45;97;97;97;98;97;45;99;97;105;48;48;48;49}  } )"


dfx canister --network ic call nft1 getTokens
```

This is one line (dfx… })"

This metadata is encoded for acceptance by a particular wallet - to be tested. Often it is the URI for the asset.

Moving into a wallet (theory):

In order for this to work, the NFT must be minted with the specific metadata and format supported by the wallet. It likely will need the canister reference URI encoded as a base64 reference. The wallet or marketplace may not accept all NFT projects.

- Install a wallet that supports NFTs (e.g. Stoic wallet)
- Go to the NFT option
- Add NFTs
- Enter the canister ID when you created it on the IC
- Boom! Done! (if the canister is in a compatible format).

not working yet.

Also, need to create an Endpoint to display it.

Appendix

ERC-1155 – Multi-NFT

- ERC1155 uses a single smart contract to represent multiple tokens at once (fungible and non-fungible).
- Storing multiple NFTs into one smart contract reduces the cost of creating a canister for each and every NFT.
- Toniq-Labs/extendable-token examples has an advanced token that enables multiple NFTs to be stored in a single canister available on GITHUB.

For the purpose of this document, building is shown using an ERC-721-like token and not the multi-NFT canister.

A word about wallets

- Wallets are a place to load, view and transfer NFTs. They will accept NFTs in formats they support.
- Popular wallets are:
  - Stoic Wallet by Toniq Labs and
  - Plug Wallet
  - EarthWallet by EarthDAO

Wallets only work with NFTs in certain formats. If it is in the wrong format, it may be possible to use a wrapper to transform one NFT format into another NFT format. For example, the metadata may need certain encoding.

A Wallet's code may be periodically updated and may result in temporary loss of access to your NFTs.