



Making an NFT on the IC

MY NOTES AS A BEGINNER IN SUPPORT OF MOTOKO BOOTCAMP



@tedreinhardt

Ver 1.0
24 Feb 2022

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.



Attribution- 4.0 International (CC BY-NC 4.0)

See <https://creativecommons.org/licenses/by-nc/4.0/legalcode>

Copyright © 2022 by T. Reinhardt

Aim

3

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

Disclaimer

- ▶ These are my notes and may contain inaccuracies.
- ▶ Use the information at your own risk. No warranty is implied.
- ▶ Sorry, I cannot provide technical support.

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

My first NFT

5

- ▶ I started learning about NFTs in August/2021. 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>

Non-Fungible Token (NFT)

6

- ▶ 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

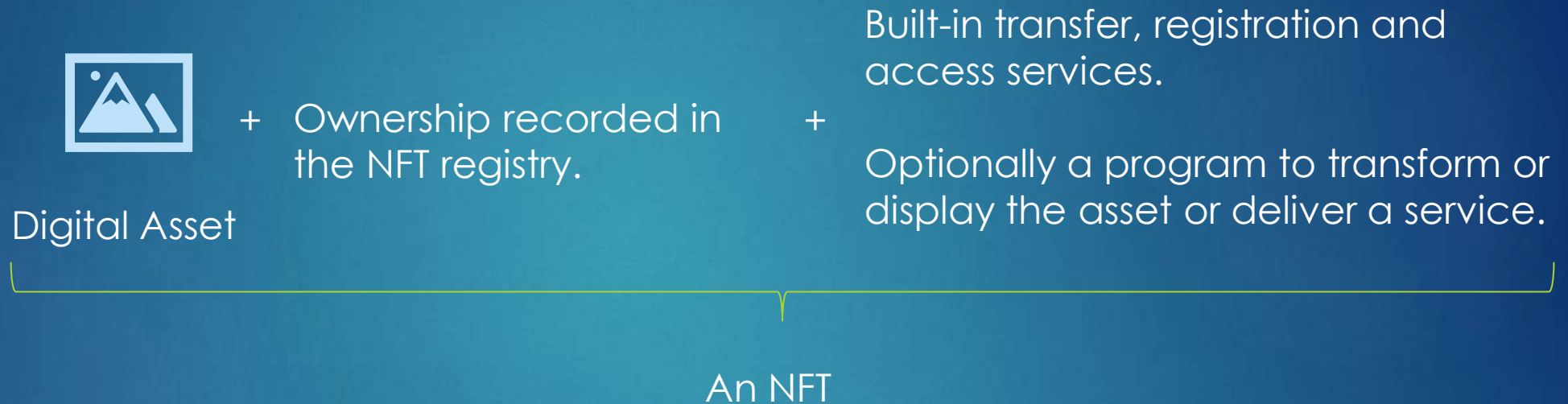
The NFT terminology

7

- ▶ 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?



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

Costs of minting an NFT on the IC

9

- ▶ 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.

10

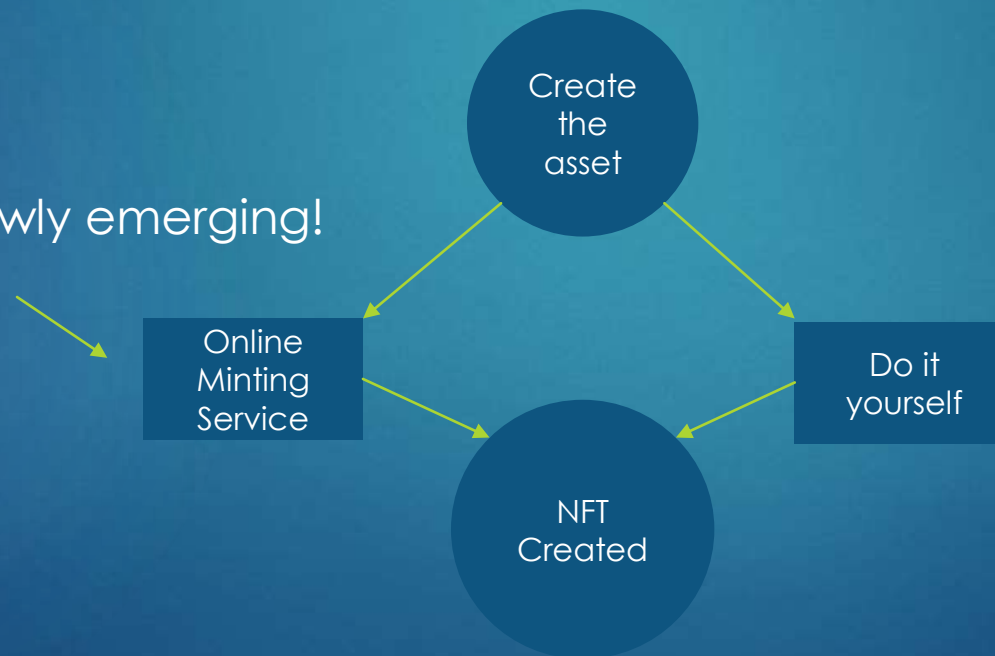
- ▶ Using a multi-NFT canister, minting 100 000 tokens would cost around \$3 vs \$300 000.

How do you make an NFT?

11

- ▶ 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

12

- ▶ 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

13

- ▶ 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?

14

- ▶ 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.

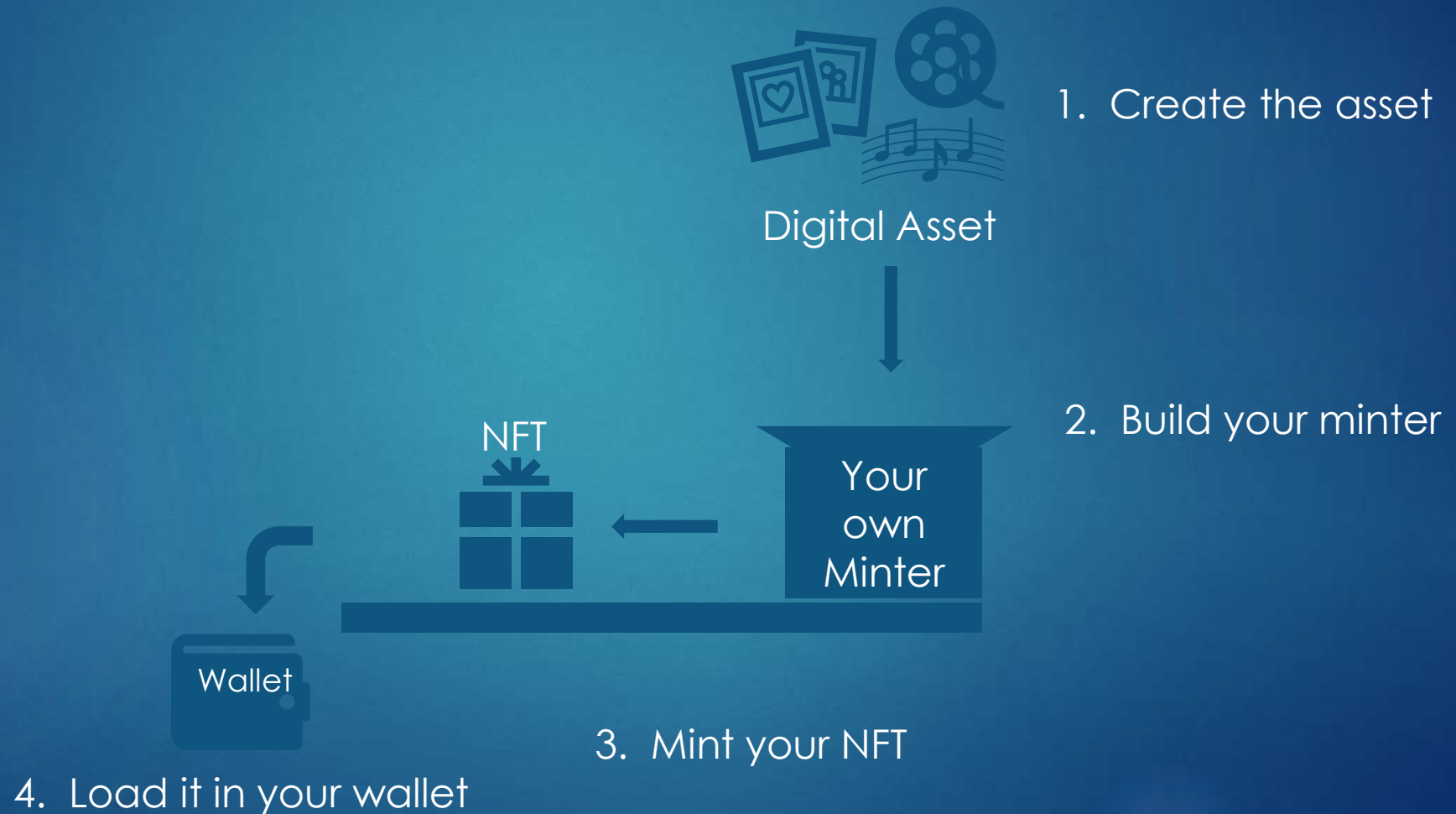
- ▶ A standard called ERC-721 (ERC is Ethereum Request for Comment) defines data elements and methods that must be present when constructing an NFT for trading within the Ethereum eco-system to aid interoperability with wallets and ecosystems.
- ▶ The Internet Computer community has not defined a standard.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

Significant NFT development work			
Name	Creator	Language	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/DepartureLabs/IC-non-fungible-token
Legends NFT	Sagacards	Motoko	https://github.com/sagacards/legends-nft

We use one derived from DIP721.

The workflow

16



Minting with a @BlocksEditor DIP721 example

DIP721 Example

18

- ▶ 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)

19

- ▶ Create a new project called dip721

```
$ dfx new dip721
```

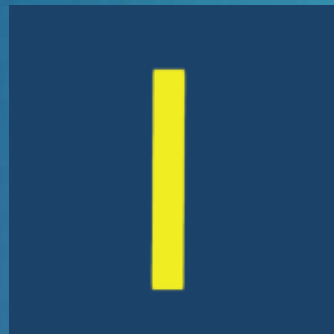
- ▶ It will install and when successful displays the following.



DIP721 - Create your digital assets

20

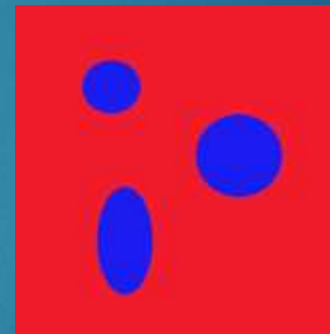
- ▶ Create 3 digital assets and store them in into the `~/dip721/src/dip721_assets/assets` subdirectory naming them `1.png`, `2.png` and `3.png`.



1.png



2.png



3.png

Deploy the canister locally and test

21

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

22

- ▶ 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



```
Compiled Output

1  import Error "mo:base/Error";
2  import Hash "mo:base/Hash";
3  import HashMap "mo:base/HashMap";
4  import Map "mo:base/Map";
5  import Option "mo:base/Option";
6  import Principal "mo:base/Principal";
7  import Array "mo:base/Array";
8  import Iter "mo:base/Iter";
9  import P "mo:base/Prelude";
10
11  actor Dip721NFT {
12    public shared query (doDown_msg) func doDown(tokenId : Nat) {
13      let caller = doDown_msg.caller; // Direct Input
14      _ownerOf(tokenId) == ?caller;
15    };
16
17    stable var name_ : Text = "ExampleRFT";
18
19    stable var symbol_ : Text = "ENFT";
20
21    // Adapted From: https://github.com/Sudden10Haze1/DIP721/blob/
22
23    private type TokenAddress = Principal;
24    private type TokenId = Nat;
25
26    private stable var tokenPk : Nat = 0;
```

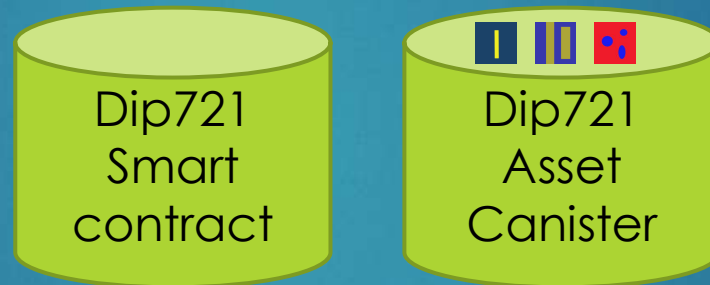
main.mo

Deploy the smart contract

23

- ▶ dfx deploy
- ▶ npm start

Results in 2 canisters



Let's try it out.

24

Show the name of the NFT

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


Show the name of the Symbol

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

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 Uniform Resource Identifier where the asset is stored



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",  
)
```

This is the TokenId



Transfer the token to someone else.

25

Confirm you own it.

Transfer it.

Confirm you don't own it.

Who is the owner now?

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

```
$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)
```

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

```
$dfx canister call dip721 ownerOf '(1)'  
  
(  
  opt principal "zht7g-jivec-azc2g-f5bkj-oxsfc-nvyo6-ch4jb-el2tb-ebyuf-zeo4d-gae",  
)
```

TokenID 1



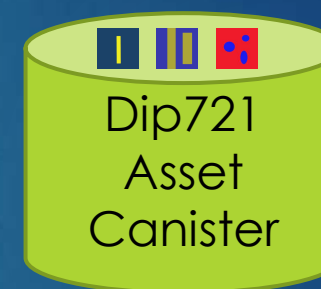
@BlocksEditor DIP721 has more methods

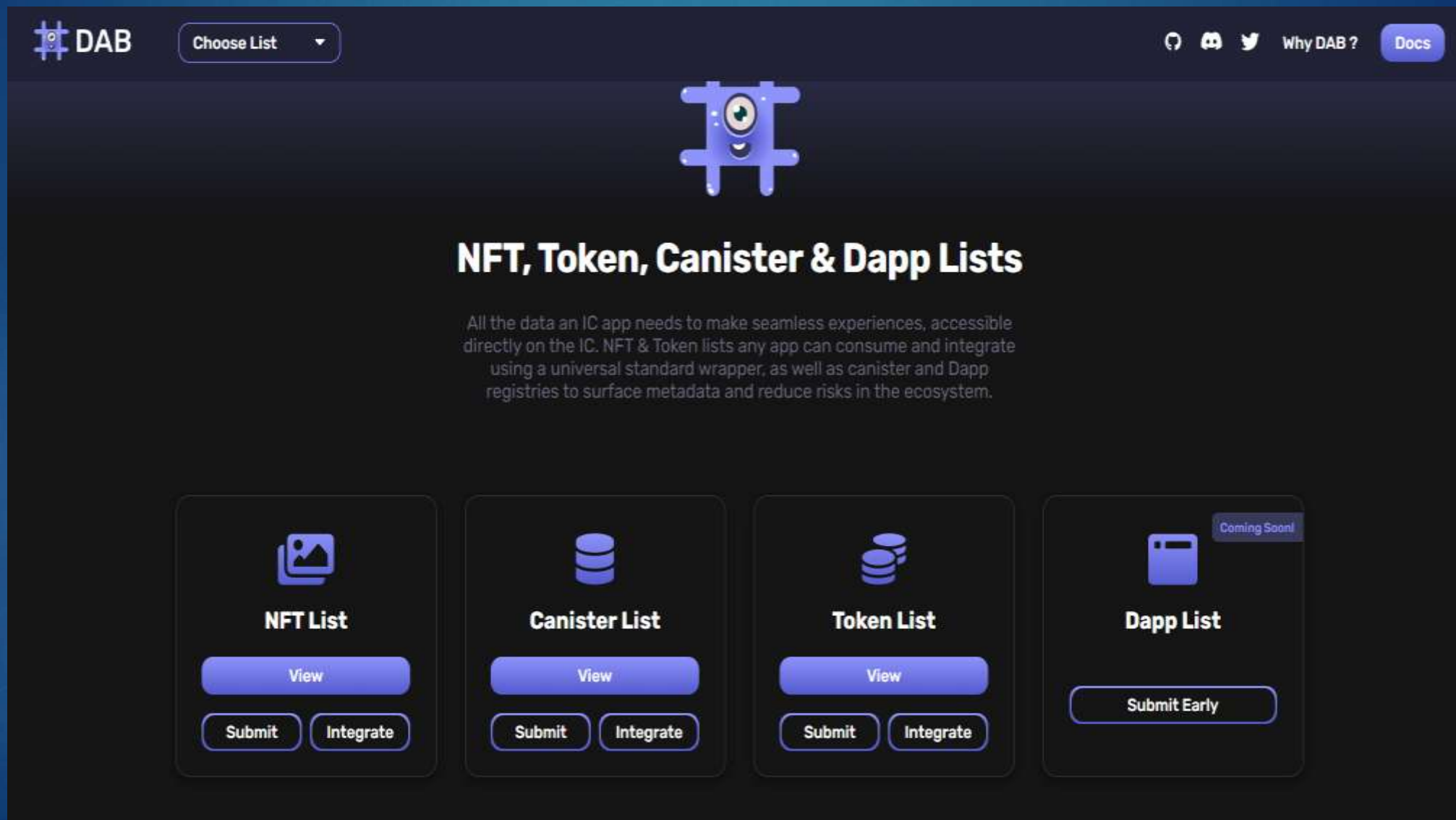
- ▶ `approve(to : Principal, tokenId : TokenId)`
- ▶ `balanceOf(p : Principal)`
- ▶ `doOwn(tokenId : Nat)`
- ▶ `getApproved(tokenId : Nat)`
- ▶ `isApprovedForAll(owner : Principal, operator : Principal)`
- ▶ `mint(uri : Text)`
- ▶ `name()`
- ▶ `ownerOf(tokenId : TokenId)`
- ▶ `setApprovalForAll(op :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

This is not production ready yet.

27

- ▶ No front end.
- ▶ No Marketplace integration.
- ▶ Lacks wallet integration.
- ▶ Access control and security is missing.
- ▶ 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.
- ▶ Payment/refund flows are missing.
- ▶ Topping/monitoring up canister cycles is missing.






The screenshot shows the DAB website interface. At the top left is the DAB logo (a purple hashtag with a camera icon) and a "Choose List" dropdown menu. On the top right are social media icons for GitHub, Discord, and Twitter, followed by links for "Why DAB?" and "Docs". In the center is a large purple icon of a camera with a face. Below this is the main heading "NFT, Token, Canister & Dapp Lists" and a paragraph explaining that DAB provides all the data an IC app needs to make seamless experiences, accessible directly on the IC. Below the text are four cards representing different list types: NFT List, Canister List, Token List, and Dapp List. Each card has a "View" button and "Submit" and "Integrate" buttons. The Dapp List card also has a "Submit Early" button and a "Coming Soon!" badge.

DAB Choose List

Why DAB ? Docs

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.



NFT List

View


Submit Integrate



Canister List

View


Submit Integrate



Token List

View

Submit Integrate



Dapp List

Submit Early

Coming Soon!

- ▶ 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

30

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

Additional info on the EXT follows.

EXT Standard

32

- ▶ This next section is based on the EXT style --- there are no standards yet on the IC.
- ▶ I have not been able to integrat into a wallet...

Steps to build (after you installed the SDK)

33

- ▶ 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 0.9.4



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 nft1
dfx help
dfx new --help
=====
```

Clone the NFT code repository from GITHUB

34

- ▶ 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

35

- ▶ 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

36

In a separate window/terminal session.

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

37

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

Build the wasm code

38

► `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/AccountIdIdentifier.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

39

- ▶ `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

40

- ▶ 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

41

- ▶ 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

42

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

One line

```
$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

44

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

This is what was achieved

45

- ▶ 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

46

- ▶ 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

48

- ▶ 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;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):

49

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

51

- ▶ 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.
- ▶ Tonic-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

52

- ▶ Wallets are a place to load, view and transfer NFTs. They will accept NFTs only 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.