**Create a dynamic NFTs collection**

**SVG NFT**

Ok, we've gained lots of context and understand about data storage in general and the benefits of SVGs specifically. Let's begin creating our very own dynamic MoodNFT with its SVG art stored on-chain.

At the core of the NFT we'll build is a flipMood function which allows the owner to flip their NFT between happy and sad images.

Start with creating the file src/MoodNft.sol and filling out the usual boilerplate. We're definitely getting good at this by now.

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.18;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";

contract MoodNft is ERC721 {

constructor() ERC721("Mood NFT", "MN"){}

}

Looking good! We want to store the SVG art on chain, we're actually going to pass these to our constructor on deployment.

constructor(string memory sadSvg, string memory happySvg) ERC721("Mood NFT", "MN"){}

We know we'll need a tokenCounter, along with this let's declare our sadSvg and happySvg as storage variables as well. All together, before getting into our functions, things should look like this:

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.18;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";

contract MoodNft is ERC721 {

string private s\_sadSvg;

string private s\_happySvg;

uint256 private s\_tokenCounter;

constructor(string memory sadSvg, string memory happySvg) ERC721("Mood NFT", "MN"){

s\_tokenCounter = 0;

s\_sadSvg = sadSvg;

s\_happySvg = happySvg;

}

}

Now we need a mint function, anyone should be able to call it, so it should definitely be public. This shouldn't be anything especially new to us so far.

function mintNft() public {

\_safeMint(msg.sender, s\_tokenCounter);

s\_tokenCounter++;

}

And now the moment of truth! As we write the tokenURI function, we know this is what defines what our NFT looks like and the metadata associated with it. Remember that we'll need to override this virtual function of the ERC721 standard.

function tokenURI(uint256 tokenId) public view override returns (string memory){}

**Wrap Up**

Our on-chain, dynamic, SVG NFT is slowly coming to life! In the next lesson, let's walk through the contents of our tokenURI function and how we can encode our SVGs in a way such that they can be reasonably stored on the blockchain.

See you there!

**Updates:**

**OpenZeppelin Updates**

*Last updated on May 31, 2024*

OpenZeppelin has replaced '\_isApprovedOrOwner' with '\_isAuthorized'

In the flipMood function '\_checkAuthorized' should be used instead of '\_isApprovedOrOwner'

Adjusted Code:

function flipMood(uint256 tokenId) public {

// Fetch the owner of the token

address owner = ownerOf(tokenId);

// Only want the owner of NFT to change the mood.

\_checkAuthorized(owner, msg.sender, tokenId);

if (s\_tokenIdToMood[tokenId] == Mood.HAPPY) {

s\_tokenIdToMood[tokenId] = Mood.SAD;

} else {

s\_tokenIdToMood[tokenId] = Mood.HAPPY;

}

}