# 666.https://stackoverflow.com/questions/67018238/which-information-hold-an-nft

**T:**Which information hold an NFT?

**Q:**I am developing an NFT market using solidity, specifically I am creating my own smart contract on top of OpenZeppelin's ERC-721 smart contract. My NFT at the moment have 5 attributes (id, image, description, collection and image) for the image, I save the hash that ipfs develve when uploading it.  
  
My question is where to save all these attributes, since I have the Image struct that has the aforementioned attributes, I add it to an array and I mint the NFT using the id of the Image object in the array and the address of the creator. I mean, I'm saving all the information outside of the ERC-721 contract, so I don't quite understand what an NFT is, since the attributes are not from the NFT but the NFT is an attribute of my struct.  
  
Am I implementing it correctly and the ERC-721 standard is only the necessary functions of an NFT or am I saving the information where it does not touch?  
  
My code is currently the following:  
  
pragma solidity ^0.5.0;import "./ERC721Full.sol";contract NftShop is ERC721Full { string public name; Image[] public nft; uint public imageId = 0; mapping(uint => bool) public \_nftExists; mapping(uint => Image) public images; struct Image { uint id; //id of the nft string hash; //hash of the ipfs string description; //nft description string collection; //what collection the nft bellongs address payable author; //creator of the nft } //Event used when new Token is created event TokenCreated( uint id, string hash, string description, string collection, address payable author ); constructor() public payable ERC721Full("NftShop", "NFTSHOP") { name = "NftShop"; } //uploadImage to the blockchain and mint the nft. function uploadImage(string memory \_imgHash, string memory \_description, string memory \_collection) public { // Make sure the image hash exists require(bytes(\_imgHash).length > 0); // Make sure image description exists require(bytes(\_description).length > 0); // Make sure collectionage exists require(bytes(\_collection).length > 0); // Make sure uploader address exists require(msg.sender!=address(0)); // Increment image id imageId ++; // Add Image to the contract images[imageId] = Image(imageId, \_imgHash, \_description, \_collection, msg.sender); //Mint the token require(!\_nftExists[imageId]); uint \_id = nft.push(images[imageId]); \_mint(msg.sender, \_id); \_nftExists[imageId] = true; // Trigger an event emit TokenCreated(imageId, \_imgHash, \_description, \_collection, msg.sender); }}   
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Any suggestions on how to improve the code if there is something weird is welcome.  
  
I hope it is not an absurd question, I am starting in the world of Ethereum.  
  
Thanks a lot.

1 **Answer**

**A1:**Yes the information such as image address and so on is stored in a contract that inherits from ERC721. The ERC721 mainly tracks ownership, minting and transfer of a token.  
  
One thing you might want to think about is the uniqueness of a token. In your example many tokens can exist that have the same image and description.You might want to prevent that by storing a hash of the \_imgHash, \_description, \_collection and require it to be unique, so that no user can create a "copy" of an existing token.  
  
something like:  
  
keccak256(abi.encodePacked(\_imgHash, \_description, \_collection))  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
another frequently used standart is the Opensea Metadata Standard, where a tokenURI is stored on the chain and the metadata lives outside the blockchain. https://docs.opensea.io/docs/metadata-standards

**C1:**Thanks for your your answer, I have a similar question, in the case of storing hash of the \_imgHash, \_description, etc... do you think this could accomodate a "royalties" scenario where each subsequent sale would generate some revenue to the creator? thanks !

**C2:**@DanielVieira An NFT as in ERC721 have one owner. When it is sold it belongs to the new owner. In a royalties scenario you'd probably want to have many owners. each sale generating royalties towards the creator of the token. That is possible but the functionality would also have to be implemented (or already has been in another standart) in a contract inheriting from an NFT such as ERC721.

**C3:**thanks so much @ruff09 , have you dabbled with the ERC-1155 standard, apparently it implements a royalty standard, seems interesting...