# 51.https://stackoverflow.com/questions/73271419/how-to-write-a-burn-function-in-smart-contract-for-nft

**T:**How to write a burn function in smart contract for NFT?

**Q:**I am learning how to write a smart contract for NFT collections and the following is the example function given by the tutorial I read:  
  
 function \_burn(uint256 tokenId) internal virtual override { super.\_burn(tokenId); if (bytes(\_tokenURIs[tokenId]).length != 0) { delete \_tokenURIs[tokenId]; } }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
I recognise that this function will not remove the token from the blockchain entirely. Instead, it will remove the URI of the token (regardless of who owns it). As a result, the token will still in the collection and be displayed on trading platforms, but the metadata will be gone (but it may take time to be in effect as platforms are not refreshing metadata frequently).  
  
I wonder if this is the right practice for burn function. It would be greatly helpful for me if someone can provide me an example of how burn function is achieved on other NFT smart contracts.

1 **Answer**

**A1:**Here is the easiest way to add burn function to an NFT.  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
 ● GO to Openzepplin Wizard  
  
 ● Click ERC721  
  
 ● Give your token a name and symbol.  
  
 ● Click on mintable and burnable, and you would get mintable and burnable NFT token contract.  
  
Here is a sample:  
  
// SPDX-License-Identifier: MITpragma solidity ^0.8.4;import "@openzeppelin/contracts/token/ERC721/ERC721.sol";import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Burnable.sol";import "@openzeppelin/contracts/access/Ownable.sol";contract MyToken is ERC721, ERC721Burnable, Ownable { constructor() ERC721("MyToken", "MTK") {} function safeMint(address to, uint256 tokenId) public onlyOwner { \_safeMint(to, tokenId); }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
The corresponding OZ wizard interface would look like this:  
  
  
  
You would get the following public burn function:  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
  
  
From Openzepplin ERC721 burnable contract  
  
You can make the contract both enumarable and burnable:  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
// SPDX-License-Identifier: MITpragma solidity ^0.8.4;import "@openzeppelin/contracts/token/ERC721/ERC721.sol";import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol";import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Burnable.sol";import "@openzeppelin/contracts/access/Ownable.sol";contract MyToken is ERC721, ERC721Enumerable, ERC721Burnable, Ownable { constructor() ERC721("MyToken", "MTK") {} function safeMint(address to, uint256 tokenId) public onlyOwner { \_safeMint(to, tokenId); } // The following functions are overrides required by Solidity. function \_beforeTokenTransfer(address from, address to, uint256 tokenId) internal override(ERC721, ERC721Enumerable) { super.\_beforeTokenTransfer(from, to, tokenId); } function supportsInterface(bytes4 interfaceId) public view override(ERC721, ERC721Enumerable) returns (bool) { return super.supportsInterface(interfaceId); }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

**C1:**So that's a built-in function. BTW, I am using ERC721Enumerable in my contract which conflicts with ERC721Burnable, but I can also see that \_burn is a built-in function inside ERC721. Maybe I can just copy that function inside ERC721Burnable into my smart contract to make it work?

**C2:**@TerryWindwalker See the update.

**C3:**Thanks! I just tried the burn function out and I see the NFT just simply disappeared.