# 3.https://stackoverflow.com/questions/72294104/check-owner-is-msg-sender-but-still-gives-erc721-transfer-caller-is-not-owner

**T:**Check owner is msg.sender, but still gives ERC721: transfer caller is not owner nor approved

**Q:**I can't seem to figure out what is wrong, even looking at other questions posted.The token should be holding the nft  
  
In the NFT contract,  
  
function NFTMe(string memory tokenUR) public returns (uint256) { uint256 newTokenId = \_tokenIds.current(); \_mint(msg.sender, newTokenId); \_setTokenURI(newTokenId, tokenUR); // setApprovalForAll(msg.sender, true); <-- this will give me reverted with reason string 'ERC721: approve to caller' \_tokenIds.increment(); return newId ; }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
I have another function to see who is the owner:  
  
function getOwner(uint256 tokenId) public view returns(address) { address owner = ERC721.ownerOf(tokenId); return owner; }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
When I deploy, it says currentOwner is the owner who minted it.  
  
In the token,  
  
constructor( string memory \_name, string memory \_symbol, address nftAddr) ERC20(\_name, \_symbol) {nft = IERC721(nftAddr);}function transfer() public { // 'ERC721: transfer caller is not owner nor approved' nft.transferFrom(msg.sender, address(this), nftId); }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

**C1:**Can you describe the expected result? Is your intention to mint the token to the user (in the NFTMe() function), and then transfer it from the user to the token contract (in the transfer() function)? Or something else?

**C2:**@PetrHejda yes correct.

1 **Answer**

**A1:**// 'ERC721: transfer caller is not owner nor approved'nft.transferFrom(msg.sender, address(this), nftId);   
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
In this case, the "transfer caller" is your token contract. Which is not the token owner, not approved by the user (the token owner) to transfer the token.  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
The easiest solution is to make the token contract automatically approved by the user from within the NFTMe() function in the NFT contract:  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
\_approve(tokenContractAddress, newTokenId);  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

**C1:**But in the token contract, it needs the input of the NFT token Id. If you look at the constructor of the token, it passes the NFT address. Seems like a chicken and egg?

**C2:**@hammies You can set the token address in the NFT contract through a custom setter function after the token contract has been deployed... So the flow can be for example: 1) deploy NFT with empty token address value, 2) deploy token passing it the NFT address, 3) set the token address in NFT.

**C3:**That makes a lot more sense!!