# 272.https://stackoverflow.com/questions/71900260/transferring-ownership-of-nfterc721

**T:**Transferring ownership of NFT(ERC721)

**Q:**I'm trying to change the ownership of a requested ERC721(NFT) to another wallet.  
  
I'm using the Mumbai Test Net and ethers.  
  
To do it I need to create the general ERC721 contract using its abi but I can't find it.  
  
In comparison, the ERC20's abi I found easily - link and used it like followed and it's working:  
  
const createERC20Contract = (contractAddress) => { const provider = new ethers.providers.Web3Provider(ethereum); const signer = provider.getSigner(); const ERC20Contract = new ethers.Contract(contractAddress, ERC20ABI, signer); return ERC20Contract;}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Where can I find the abi of ERC721 to create the ethers.Contract() ?

1 **Answer**

**A1:**Here's a generic ERC721 ABI that I built from the OpenZeppelin implementation. (The bare minimum contract - none of their extensions such as mint and burn tokens.)  
  
[ { "inputs": [ { "internalType": "string", "name": "name\_", "type": "string" }, { "internalType": "string", "name": "symbol\_", "type": "string" } ], "stateMutability": "nonpayable", "type": "constructor" }, { "anonymous": false, "inputs": [ { "indexed": true, "internalType": "address", "name": "owner", "type": "address" }, { "indexed": true, "internalType": "address", "name": "approved", "type": "address" }, { "indexed": true, "internalType": "uint256", "name": "tokenId", "type": "uint256" } ], "name": "Approval", "type": "event" }, { "anonymous": false, "inputs": [ { "indexed": true, "internalType": "address", "name": "owner", "type": "address" }, { "indexed": true, "internalType": "address", "name": "operator", "type": "address" }, { "indexed": false, "internalType": "bool", "name": "approved", "type": "bool" } ], "name": "ApprovalForAll", "type": "event" }, { "anonymous": false, "inputs": [ { "indexed": true, "internalType": "address", "name": "from", "type": "address" }, { "indexed": true, "internalType": "address", "name": "to", "type": "address" }, { "indexed": true, "internalType": "uint256", "name": "tokenId", "type": "uint256" } ], "name": "Transfer", "type": "event" }, { "inputs": [ { "internalType": "address", "name": "to", "type": "address" }, { "internalType": "uint256", "name": "tokenId", "type": "uint256" } ], "name": "approve", "outputs": [], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [ { "internalType": "address", "name": "owner", "type": "address" } ], "name": "balanceOf", "outputs": [ { "internalType": "uint256", "name": "", "type": "uint256" } ], "stateMutability": "view", "type": "function" }, { "inputs": [ { "internalType": "uint256", "name": "tokenId", "type": "uint256" } ], "name": "getApproved", "outputs": [ { "internalType": "address", "name": "", "type": "address" } ], "stateMutability": "view", "type": "function" }, { "inputs": [ { "internalType": "address", "name": "owner", "type": "address" }, { "internalType": "address", "name": "operator", "type": "address" } ], "name": "isApprovedForAll", "outputs": [ { "internalType": "bool", "name": "", "type": "bool" } ], "stateMutability": "view", "type": "function" }, { "inputs": [], "name": "name", "outputs": [ { "internalType": "string", "name": "", "type": "string" } ], "stateMutability": "view", "type": "function" }, { "inputs": [ { "internalType": "uint256", "name": "tokenId", "type": "uint256" } ], "name": "ownerOf", "outputs": [ { "internalType": "address", "name": "", "type": "address" } ], "stateMutability": "view", "type": "function" }, { "inputs": [ { "internalType": "address", "name": "from", "type": "address" }, { "internalType": "address", "name": "to", "type": "address" }, { "internalType": "uint256", "name": "tokenId", "type": "uint256" } ], "name": "safeTransferFrom", "outputs": [], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [ { "internalType": "address", "name": "from", "type": "address" }, { "internalType": "address", "name": "to", "type": "address" }, { "internalType": "uint256", "name": "tokenId", "type": "uint256" }, { "internalType": "bytes", "name": "\_data", "type": "bytes" } ], "name": "safeTransferFrom", "outputs": [], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [ { "internalType": "address", "name": "operator", "type": "address" }, { "internalType": "bool", "name": "approved", "type": "bool" } ], "name": "setApprovalForAll", "outputs": [], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [ { "internalType": "bytes4", "name": "interfaceId", "type": "bytes4" } ], "name": "supportsInterface", "outputs": [ { "internalType": "bool", "name": "", "type": "bool" } ], "stateMutability": "view", "type": "function" }, { "inputs": [], "name": "symbol", "outputs": [ { "internalType": "string", "name": "", "type": "string" } ], "stateMutability": "view", "type": "function" }, { "inputs": [ { "internalType": "uint256", "name": "tokenId", "type": "uint256" } ], "name": "tokenURI", "outputs": [ { "internalType": "string", "name": "", "type": "string" } ], "stateMutability": "view", "type": "function" }, { "inputs": [ { "internalType": "address", "name": "from", "type": "address" }, { "internalType": "address", "name": "to", "type": "address" }, { "internalType": "uint256", "name": "tokenId", "type": "uint256" } ], "name": "transferFrom", "outputs": [], "stateMutability": "nonpayable", "type": "function" }]  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

**C1:**Thanks for the fast answer :) It's now working with 'transferFrom' and not 'safeTransferFrom'. But it is good enough for now. What is the best practice for transferring ownership of NFT? To transfer it on the client-side using like I've shown here? or use the IERC721 in the .sol file?

**C2:**@AlonBenYaakov It depends on your use case. Mind that for client-side transfer, you'll need the token owner to sign a transaction performing the transfer (e.g. through their MetaMask wallet). For a transfer through a third-party contract, your contract needs to be an owner or an operator of this NFT (the user needs to approve() your contract to operate their token).