# 471.https://stackoverflow.com/questions/70811644/buy-nft-from-wallet1-which-has-all-minted-nft-to-the-user-wallet-connected-with

**T:**Buy NFT from wallet1 which has all minted nft to the user wallet connected with phantom, Javascript, Solana

**Q:**Here is my code to transfer nft from Alice Account (which has my all minted nfts) and feePayer account which will buy from the Alice Wallet. Well this works perfectly fine like this. But the problem is when the same transection I try to sign and send with phantom wallet it gives me an error (You can check it by the commented line 101 to 108) when I uncomment the phantom signAndSend method code and try to execute it, it gives me error RPC req rejected but the same is successfully done with the connection.sendTransaction.  
  
That was problem 1,Now problem 2 is thatIn this case, I had used feePayer which had the keypair already, but if I want to make it as user connects with connectWallet button, ill had only publicKey of the user. Now in that situaion how can i make the the nft transfer from Alice wallet to the user who connected and clicked on buy?  
  
JS Code:  
  
const sendNft = async () => {const feePayer = Keypair.fromSecretKey( bs58.decode( "3DdVyZuANr5en2PQymCPmoFBMsfdhjaRHqnk3ejW\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" ));// G2FAbFQPFa5qKXCetoFZQEvF9BVvCKbvUZvodpVidnoYconst alice = Keypair.fromSecretKey( bs58.decode( "2YQDdnfxiHPKu9GypLX1yXaQTQojvDSPgFkDxrU\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" ));const mintPubkey = new PublicKey( "A8SJfwzKJAaMrY6Lb9FxZCfmVMVLcjKvRuzAiNiU6of5");const getProvider = async () => { if ("solana" in window) { // opens wallet to connect to await window.solana.connect(); const provider = window.solana; if (provider.isPhantom) { console.log("Is Phantom installed? ", provider.isPhantom); return provider; } } else { window.open("https://www.phantom.app/", "\_blank"); }};window.solana.on("connect", () => console.log("connected!"))const network = "https://api.devnet.solana.com";const connection = new Connection(network);var prodivder = await getProvider();console.log(prodivder.publicKey.toString())var provider = await getProvider()console.log(provider)// calculate ATAlet accountAlice = await Token.getAssociatedTokenAddress( ASSOCIATED\_TOKEN\_PROGRAM\_ID, // always ASSOCIATED\_TOKEN\_PROGRAM\_ID TOKEN\_PROGRAM\_ID, // always TOKEN\_PROGRAM\_ID mintPubkey, // mint alice.publicKey // owner);console.log(`ATA: ${accountAlice.toBase58()}`);let tx3 = new Transaction().add( Token.createAssociatedTokenAccountInstruction( ASSOCIATED\_TOKEN\_PROGRAM\_ID, // always ASSOCIATED\_TOKEN\_PROGRAM\_ID TOKEN\_PROGRAM\_ID, // always TOKEN\_PROGRAM\_ID mintPubkey, // mint accountAlice, // ata alice.publicKey, // owner of token account alice.publicKey // fee payer ));// calculate ATAlet accountPayer = await Token.getAssociatedTokenAddress( ASSOCIATED\_TOKEN\_PROGRAM\_ID, // always ASSOCIATED\_TOKEN\_PROGRAM\_ID TOKEN\_PROGRAM\_ID, // always TOKEN\_PROGRAM\_ID mintPubkey, // mint feePayer.publicKey // owner);console.log(`ATA: ${accountPayer.toBase58()}`);let tx2 = new Transaction().add( Token.createAssociatedTokenAccountInstruction( ASSOCIATED\_TOKEN\_PROGRAM\_ID, // always ASSOCIATED\_TOKEN\_PROGRAM\_ID TOKEN\_PROGRAM\_ID, // always TOKEN\_PROGRAM\_ID mintPubkey, // mint accountPayer, // ata feePayer.publicKey, // owner of token account feePayer.publicKey // fee payer ));// console.log(`txhash: ${await connection.sendTransaction(tx2, [feePayer])}`);let tx = new Transaction().add( Token.createTransferCheckedInstruction( TOKEN\_PROGRAM\_ID, // always TOKEN\_PROGRAM\_ID accountAlice, // from (should be a token account) mintPubkey, // mint accountPayer, // to (should be a token account) alice.publicKey, // owner of from [], // for multisig account, leave empty. 1, // amount, if your deciamls is 8, send 10^8 for 1 token 0 // decimals ));///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////// Commenting out this, thus is the phantom codestx2.recentBlockhash = (await connection.getRecentBlockhash()).blockhash;tx2.feePayer = feePayer.publicKey;const { signature } = await window.solana.signAndSendTransaction(tx2);await connection.confirmTransaction(signature);// phantom code ends console.log(`txhash: ${await connection.sendTransaction(tx, [feePayer, alice /\* fee payer + owner \*/])}`);};  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

0 **Answer**