# 596.https://stackoverflow.com/questions/69475970/how-can-we-use-phantom-wallet-to-mint-nft-in-solana-without-using-minter-account

**T:**how can we use phantom wallet to mint NFT in solana without using minter account

**Q:**I have been trying to mint NFT in solana. Have done it in different waysMostly every way I tried uses random generated keyPair to first mint tokens and then transfer to our walletIs there any way I can mint using my phantom wallet  
  
In the following code minting is performed By minter account that is generated using web3 Keypair.generate functionIs there any way I can perform this minting directly using phantom wallet  
  
// The code that I tried last is  
  
import \* as web3 from '@solana/web3.js';import \* as splToken from '@solana/spl-token'; const getProvider = async () => { if ("solana" in window) { const provider = window.solana; if (provider.isPhantom) { console.log("Is Phantom installed? ", provider.isPhantom); return provider; } } else { window.open("https://www.phantom.app/", "\_blank"); } };const mintingTest = async () => { const phantomProvider = await getProvider(); const mintRequester = await phantomProvider.publicKey; console.log("Public key of the mint Requester: ", mintRequester.toString()); //To connect to the mainnet, write mainnet-beta instead of devnet const connection = new web3.Connection( web3.clusterApiUrl('devnet'), 'confirmed', ); //This fromWallet is your minting wallet, that will actually mint the tokens var fromWallet = web3.Keypair.generate(); // Associate the mintRequester with this wallet's publicKey and privateKey // This is basically the credentials that the mintRequester (creator) would require whenever they want to mint some more tokens // Testing the parameters of the minting wallet console.log("Creator's Minting wallet public key: ",fromWallet.publicKey.toString()); console.log(fromWallet.secretKey.toString()); // Airdrop 1 SOL to the minting wallet to handle the minting charges var fromAirDropSignature = await connection.requestAirdrop( fromWallet.publicKey, web3.LAMPORTS\_PER\_SOL, ); await connection.confirmTransaction(fromAirDropSignature); console.log("Airdropped (transferred) 1 SOL to the fromWallet to carry out minting operations"); // This createMint function returns a Promise <Token> let mint = await splToken.Token.createMint( connection, fromWallet, fromWallet.publicKey, null, 6, // Number of decimal places in your token splToken.TOKEN\_PROGRAM\_ID, ); // getting or creating (if doens't exist) the token address in the fromWallet address // fromTokenAccount is essentially the account \*inside\* the fromWallet that will be able to handle the new token that we just minted let fromTokenAccount = await mint.getOrCreateAssociatedAccountInfo( fromWallet.publicKey, ); // getting or creating (if doens't exist) the token address in the toWallet address // toWallet is the creator: the og mintRequester // toTokenAmount is essentially the account \*inside\* the mintRequester's (creator's) wallet that will be able to handle the new token that we just minted let toTokenAccount = await mint.getOrCreateAssociatedAccountInfo( mintRequester, ); // // Minting 1 token await mint.mintTo( fromTokenAccount.address, fromWallet.publicKey, [], 1000000 // 1 followed by decimals number of 0s // You'll ask the creator ki how many decimals he wants in his token. If he says 4, then 1 token will be represented as 10000 ); console.log("Initial mint successful"); // This transaction is sending of the creator tokens(tokens you just created) from their minting wallet to their Phantom Wallet var transaction = new web3.Transaction().add( splToken.Token.createTransferInstruction( splToken.TOKEN\_PROGRAM\_ID, fromTokenAccount.address, toTokenAccount.address, fromWallet.publicKey, [], 1000000, // This is transferring 1 token, not 1000000 tokens ), ); var signature = await web3.sendAndConfirmTransaction( connection, transaction, [fromWallet], {commitment: 'confirmed'}, ); const creatorTokenAddress = mint.publicKey; const creatorTokenAddressString = mint.publicKey.toString(); console.log("SIGNATURE: ", signature); //Signature is basically like the paying party signs a transaction with their key. console.log("Creator Token Address: ", creatorTokenAddressString); console.log("Creator Minting Wallet Address: ", mint.payer.publicKey.toString()); let creatorTokenBalance = await toTokenAccount.amount; console.log("Creator's Token Balance: ", creatorTokenBalance); };  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

**C1:**This doesn't directly answer the question, but you'll probably have an easier time with the Solana wallet adapter package, which provides easy connections to many different wallet providers, including Phantom: github.com/solana-labs/wallet-adapter

1 **Answer**

**A1:**You can now do this easily using @metaplex-foundation/js  
  
Here's a snippet  
  
import { Metaplex, token } from "@metaplex-foundation/js";import { Connection, clusterApiUrl } from "@solana/web3.js";const connection = new Connection(clusterApiUrl("mainnet-beta"));const metaplex = new Metaplex(connection) // if you have to provide the wallet identity, you can pass it here .use(walletAdapterIdentity(wallet))const task = metaplex.tokens().createTokenWithMint({ // you can supply additional parameters here // by default, any accounts with authority will use the identity given above initialSupply: token(100\_000\_000\_000),})const tokenInfo = await task.run()  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]