# 601.https://stackoverflow.com/questions/69417491/nodejsget-not-repeated-numbers-from-a-range

**T:**[NodeJS]Get not repeated numbers from a range

**Q:**I am trying to update via API some of my arts on a NFT test over OpenSea, now he updates but the problem is that he is repeating the numbers, is it possible to choose a number in a range but that never repeat?  
  
My actual code:  
  
 const opensea = require("opensea-js");const OpenSeaPort = opensea.OpenSeaPort;const Network = opensea.Network;const MnemonicWalletSubprovider = require("@0x/subproviders") .MnemonicWalletSubprovider;const RPCSubprovider = require("web3-provider-engine/subproviders/rpc");const Web3ProviderEngine = require("web3-provider-engine");const MNEMONIC = 'SECRET';const NODE\_API\_KEY = 'MyKEY';const isInfura = true;const FACTORY\_CONTRACT\_ADDRESS = '0x745e6b0CAd1eDc72647B9fFec5C69e4608f73ab2';const NFT\_CONTRACT\_ADDRESS = '0x745e6b0CAd1eDc72647B9fFec5C69e4608f73ab2';const OWNER\_ADDRESS = '0xaEBB892210eB23C47b1e710561c7BC4CFA63A62e';const NETWORK = 'rinkeby';const API\_KEY = ""; // API key is optional but useful if you're doing a high volume of requests.if (!MNEMONIC || !NODE\_API\_KEY || !NETWORK || !OWNER\_ADDRESS) { console.error( "Please set a mnemonic, Alchemy/Infura key, owner, network, API key, nft contract, and factory contract address." ); return;}if (!FACTORY\_CONTRACT\_ADDRESS && !NFT\_CONTRACT\_ADDRESS) { console.error("Please either set a factory or NFT contract address."); return;}const BASE\_DERIVATION\_PATH = `44'/60'/0'/0`;const mnemonicWalletSubprovider = new MnemonicWalletSubprovider({ mnemonic: MNEMONIC, baseDerivationPath: BASE\_DERIVATION\_PATH,});const network = NETWORK === "mainnet" || NETWORK === "live" ? "mainnet" : "rinkeby";const infuraRpcSubprovider = new RPCSubprovider({ rpcUrl: isInfura ? "https://" + network + ".infura.io/v3/" + NODE\_API\_KEY : "https://eth-" + network + ".alchemyapi.io/v2/" + NODE\_API\_KEY,});const providerEngine = new Web3ProviderEngine();providerEngine.addProvider(mnemonicWalletSubprovider);providerEngine.addProvider(infuraRpcSubprovider);providerEngine.start();const seaport = new OpenSeaPort( providerEngine, { networkName: NETWORK === "mainnet" || NETWORK === "live" ? Network.Main : Network.Rinkeby, apiKey: API\_KEY, }, (arg) => console.log(arg));async function sellTheItems() { // Example: simple fixed-price sale of an item owned by a user. console.log("Auctioning an item for a fixed price..."); //Get a number in a range and return as a string function getRandomInt(min, max) { min = Math.ceil(min); max = Math.floor(max); return Math.floor(Math.random() \* (max - min)) + min; //The maximum is exclusive and the minimum is inclusive } const fixedPriceSellOrder = await seaport.createSellOrder({ asset: { tokenId: getRandomInt(0, 500).toString(), //tokenId: "0", tokenAddress: NFT\_CONTRACT\_ADDRESS, }, startAmount: 0.36, expirationTime: 0, accountAddress: OWNER\_ADDRESS, }); console.log( `Successfully created a fixed-price sell order! ${fixedPriceSellOrder.asset.openseaLink}\n` );}async function doItAgain() { sellTheItems();}//Repeat doItAgain() every 5 secondssetInterval(doItAgain, 5000);  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
What I want to do is get a number between 0 and 500 but never repeat a number that I used before, my intention is get a number to 'tokenId' everytime.

**C1:**Please ensure to create a minimal reproducible example, otherwise it might get downvoted by the community

**C2:**Random by definition, will return a random value - which could be an old value, your only solution is to store the previously generated random values, and check the with the newly generated ones - or use something easier like the current timestamp

**C3:**Thanks for the warning mr. @SamridhTuladhar. Thanks for the explanation, if was in Python I was about to solve, but this system was made in Javascript, so I don't know much how to work with them.

1 **Answer**

**A1:**With respect, it's a bad idea to use 3-digit "random" tokens. Partly because you get duplicates and you need some sort of storage to avoid that. Partly because they're easy to brute-force.  
  
Use the uuid Universal Unique IDentifier package to generate 128-bit uuidv4 tokens of which 60 bits are random, using a crypographically secure pseudo-random number generator. All uuids, not just the uuidv4 random ones, are constructed to avoid duplicates. But the uuidv4 tokens have the advantage that they're very hard for cybercreeps to guess.  
  
const { v4: uuidv4 } = require('uuid');... tokenId: uuidv4(), // ⇨ '1b9d6bcd-bbfd-4b2d-9b5d-ab8dfbbd4bed'  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

**C1:**The NFT standard, ERC-721, requires the token ID to be a 256bit unsigned integer in the decimal form. Can you update the answer so that it passes the number in the decimal form (as a string)?

**C2:**Thanks Jones, the tokens are for Smart Contracts, so they have an ID (each token have a ID), but I need to sell them. So, to set a sell option I need to set what ID I am selling, so what I want is get the ID and increase +1 until reach the 499!