# 84.https://stackoverflow.com/questions/72992643/how-to-update-a-nft-mint-price-by-ethereum-price-in-usd

**T:**How to update a NFT mint price by ethereum price in USD?

**Q:**I am trying to build a smart contract that would give a fixed price in USD for each NFT to be minted by others, which they will need to pay in ETH. But I found a problem that the price of ETH is always changing, and each update of the NFT price in ETH would need some gas fee, which could cost a lot in long term for maintenance. Is there a way to periodically update ETH price inside the smart contract, or is manual updating the only way to do it?  
  
Or I might have to remove the NFT price limit and completely rely on the frontend to handle the pricing part. But I think that's too risky.

1 **Answer**

**A1:**You can use a Chainlink datafeed that returns the price of ETH in USD.  
  
There are no datafeeds in emulators (e.g. Ganache or the Remix IDE built-in network), so you can test this snippet on your local fork of the Ethereum mainnet.  
  
pragma solidity 0.8;import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";contract MyContract { AggregatorV3Interface priceFeed; // 18 decimals uint256 requiredPriceInUsd = 1000 \* 1e18; constructor() { // https://etherscan.io/address/0x5f4eC3Df9cbd43714FE2740f5E3616155c5b8419#code // Chainlink ETH/USD Price Feed for Ethereum Mainnet priceFeed = AggregatorV3Interface(0x5f4eC3Df9cbd43714FE2740f5E3616155c5b8419); } // returns amount of wei function getRequiredPriceInWei() public view returns (uint256) { (,int answer,,,) = priceFeed.latestRoundData(); // returned price is 8 decimals, convert to 18 decimals uint256 ethUsdPrice = uint256(answer) \* 1e10; // 36 decimals / 18 decimals = 18 decimals return (requiredPriceInUsd \* 1e18) / ethUsdPrice; }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Output from my test:  
  
 ● answer is 122884000000 (1228 USD and 8 decimals)  
  
 ● returned value from getRequiredPriceInWei() is 813775593242407473 (of wei, that's ~0.8 ETH for 1,000 USD)

**C1:**In this way, I can get the price of an NFT without paying gas. But if I am about to ensure the user has provided enough Eth in minting the NFT, it should still be stored somewhere (which would need gas), right? Or should I recalculate it every time a new minting request is created and allow it to proceed if the Eth given is within a range?

**C2:**@TerryWindwalker The second option - if sent ETH is within a range - is easier. You can build a condition that requires the sent amount to be exactly the required amount (or within a range): require(msg.value == getRequiredPriceInWei())

**C3:**I am setting a range of 99%-101% of the calculated price. Would that be enough for most cases? Or could it be too much? I think Eth's price is changing a lot these days.

**C4:**@TerryWindwalker A usual scenario is that you load the currently required wei amount from getRequiredPriceInWei() to a frontend app, and then generate a transaction with this specific amount for the user's wallet (e.g. MetaMask) to sign. You can synchronize the price generated on the web app with the Chainlink datafeed refresh interval (it's not live data - it's updating the price in some interval), so that there's lower chance of the transaction failing in case the user loads the page and sends the transaction after a day for example.