# 200.https://stackoverflow.com/questions/72433091/is-payble-keyword-neccessary-even-if-i-am-not-taking-any-ether-matic

**T:**Is payble keyword neccessary even if I am not taking any ether/matic?

**Q:** function sendGift(uint256 \_mintAmount,address recipient) public payable { uint256 supply = totalSupply(); require(!paused); require(\_mintAmount > 0); require(\_mintAmount <= maxMintAmount); require(supply + \_mintAmount<= availableSupplyForSale); //require(\_amount >= cost \* \_mintAmount); require(coinToken.allowance(msg.sender,address(this))>=cost \* \_mintAmount); coinToken.transferFrom(msg.sender, address(this),cost \* \_mintAmount); if(supply<currentSupply){ for (uint256 i = 1; i <= \_mintAmount; i++) { \_safeMint(recipient, supply + i); } } else{ uint256[] memory tokenIds = walletOfOwner(address(this)); for(uint256 i=1;i<=\_mintAmount;i++) transferFrom(address(this),recipient,tokenIds[i]); } }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Do I need to use payable here? Contract does not take any matic. It only takes custom token as payment .  
  
(bool os, ) = payable(admin).call{value: address(this).balance}(""); require(os);  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Also since I am not taking any matic, will this above line necessary for withdraw assets from contract as an owner? I have a sense that this above line only is useful to withdraw eth/polygon.  
  
I am a new blockchain kid. Please help.

1 **Answer**

**A1:**The payable modifier of a function is required when your function accepts native tokens (ETH, BNB, MATIC, ... depending on the network).  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
So in this case, you can safely remove it from the function header.  
  
// removed `payable`function sendGift(uint256 \_mintAmount,address recipient) public {  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
The low-level .call() also doesn't require using payable to send native tokens.  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
payable(admin).call{value: address(this).balance}("");  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
However, if you used the high-level .transfer(), then you'd need to cast the admin variable type address to its extension type address payable using the typecasting function.  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
// will not work as it's type `address`admin.transfer(address(this).balance);  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
// need to cast type `address` to type `address payable`payable(admin).transfer(address(this).balance);  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
In order to withdraw tokens from your contract address, you need to invoke the transfer() function (defined in the ERC-20 standard) on the token contract. Do not confuse it with the native transfer() function of address payable, these are two separate things, just with the same name.  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
interface IERC20 { function transfer(address, uint256) external returns (bool);}contract MyContract { function withdrawToken() { IERC20(tokenContractAddress).transfer(recipient, amount); }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]