# 158.https://stackoverflow.com/questions/72635880/despite-using-the-receive-function-my-contract-is-not-receiving-payment-in-re

**T:**Despite using the receive() function, my contract is not receiving payment in Remix

**Q:**I am writing a smart contract in Remix with Solidity. The purpose of the contract is to allow a user to mint an NFT for 1 ETH. At present, the user is able to mint and the contract accepts the payment (ie. the user's balance is properly subtracted). But when I check the address(this).balance of the contract with my accountBalance() function, the function returns 0. I have included the receive() function as per the Solidity docs:  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
event Received(address, uint); receive() external payable { emit Received(msg.sender, msg.value); }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Can someone explain why this is happening and what I need to change about my contract? Here is my contract:  
  
// SPDX-License-Identifier: MITpragma solidity ^0.8.4;// importsimport '@openzeppelin/contracts/token/ERC721/ERC721.sol';import '@openzeppelin/contracts/access/Ownable.sol';import '@openzeppelin/contracts/security/PullPayment.sol';// contractcontract RobocopPoster is ERC721, Ownable, PullPayment { // constants uint256 public mintPrice; uint256 public totalSupply; uint256 public maxSupply; uint256 public maxPerWallet; bool public mintEnabled; mapping (address => uint256) public walletMints; // constructor // initialize variables constructor() payable ERC721('RobocopPoster', 'SFFPC') { mintPrice = 1 ether; totalSupply = 0; maxSupply = 1000; maxPerWallet = 3; } event Received(address, uint); receive() external payable { emit Received(msg.sender, msg.value); } // functions function setMintEnabled(bool mintEnabled\_) external onlyOwner { mintEnabled = mintEnabled\_; } function withdrawPayments(address payable payee) public override onlyOwner virtual { super.withdrawPayments(payee); } function accountBalance() public view returns (uint256) { return (address(this).balance); } function mint(uint256 quantity\_) public payable { require(mintEnabled, 'Minting not enabled.'); require(msg.value == quantity\_ \* mintPrice, 'wrong mint value'); require(totalSupply + quantity\_ <= maxSupply, 'sold out'); require(walletMints[msg.sender] + quantity\_ <= maxPerWallet, 'exceed max wallet'); walletMints[msg.sender] += quantity\_; \_asyncTransfer(address(this), msg.value); for (uint i = 0; i < quantity\_; i++) { uint256 newTokenId = totalSupply + 1; totalSupply++; \_safeMint(msg.sender, newTokenId); } }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

**C1:**Remix can be a bit confusing sometimes imo. I like to use hardhat, specifically the github.com/scaffold-eth/scaffold-eth template for testing my contracts. There's a built in front end that allows you to test all your methods out

1 **Answer**

**A1:**You need to call withdrawPayments to receive the fund, because \_asyncTransfer from PullPayment in your contract minting sent the fund to the escrow contract. That's why you saw zero balance in ERC721 contract.  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

**C1:**I appreciate your input but I'm not sure I'm following the logic here correctly. In my \_asynceTransfer() function I'm setting the address to my address(this) (ie. RobocopPoster.sol). Isn't the purpose of \_asyncTransfer() to specify where the msg.value is sent?

**C2:**It is not. You don't need any function to receive ETH, just payable modifier is needed. \_asyncTransfer is actually send your ETH out to the escrow (middleman) wallet. You can try removing \_asyncTransfer and it will just work. And recommend to take time reading the contract source code of PullPayment.sol and Escrow.sol, you will understand the code a lot better :)

**C3:**Thank you very much for your help. I have reviewed the PullPayment.sol and Escrow.sol docs and I understand what you mean now. My contract is now working as expected :)