# 667.https://stackoverflow.com/questions/66959912/identifier-already-declared-for-context-sol-in-openzeppelin

**T:**Identifier already declared for Context.sol in Openzeppelin

**Q:**I am working on an NFT project. I have my NFT file in my contract folder which is to import from the openzeppelin files in nodemodules. However, the compiler seem to suggest Context.sol is already declared in my file. I checked the Context.sol file and there seem to be a lot in the folders given, how do I work around that.  
  
Error: DeclarationError: Identifier already declared.import "@openzeppelin/contracts/token/ERC721/ERC721.sol";^-------------------------------------------------------^@openzeppelin/contracts/utils/Context.sol:15:2: The previous declaration is here:contract Context {^ (Relevant source part starts here and spans across multiple lines)  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
NFT.sol  
  
pragma solidity ^0.5.0;import "@openzeppelin/contracts/access/Ownable.sol";import "@openzeppelin/contracts/token/ERC721/ERC721.sol";contract NFT is ERC721, Ownable { address payable public \_owner; mapping (uint => bool) public sold; mapping (uint => uint) public price; event Purchase(address owner, uint price, uint id, string uri); constructor() ERC721("Dapp University", "DAPPU") public{ \_owner = msg.sender; } function mint(string memory \_tokenURI, uint \_price) public onlyOwner returns (bool) { uint \_tokenId = totalSupply() + 1; price[\_tokenId] = \_price; \_mint(address(this), \_tokenId); \_setTokenURI(\_tokenId, \_tokenURI); return true; } function buy(uint \_id) external payable { \_validate(\_id); //check req. for trade \_trade(\_id); //swap nft for eth emit Purchase(msg.sender, price[\_id], \_id, tokenURI(\_id)); } function \_validate(uint \_id) internal { require(\_exists(\_id), "Error, wrong Token id"); //not exists require(!sold[\_id], "Error, Token is sold"); //already sold require(msg.value >= price[\_id], "Error, Token costs more"); //costs more } function \_trade(uint \_id) internal { \_transfer(address(this), msg.sender, \_id); //nft to user \_owner.transfer(msg.value); //eth to owner sold[\_id] = true; //nft is sold }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Context.sol  
  
pragma solidity ^0.5.0;contract Context { function \_msgSender() internal view virtual returns (address) { return msg.sender; } function \_msgData() internal view virtual returns (bytes calldata) { this; // silence state mutability warning without generating bytecode - see https://github.com/ethereum/solidity/issues/2691 return msg.data; } }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

1 **Answer**

**A1:**I found the answer. It is a problem that is going to be faced irrespective of the kind of work you are doing using @OpenZeppilin contracts. There is a Context.sol file in each of the subfolders to help with independent projects. However, this large of number of Context.sol files collide with each other throwing an error from the compiler. As such as seen from the error thrown, I had to trace back the imports and realized the Context.sol was being tracked and imported from the GSN folder instead of the Utils folder, so I rechanged the imports to take it from the utils folder as seen here.The virtual keyword also has to be cleaned up in case you are using <0.6.0 compiler(I suppose that will be a different answer to a different question altogether)  
  
pragma solidity ^0.5.0;import "@openzeppelin/contracts/utils/Context.sol";// Change to utils folder instead of GSN folder and possibly for all clashing // Context.sol files import "./IERC721.sol"; import "./IERC721Receiver.sol"; import "../../math/SafeMath.sol"; import "../../utils/Address.sol"; import "../../drafts/Counters.sol"; import "../../introspection/ERC165.sol"; contract ERC721 is Context, ERC165, IERC721 { using SafeMath for uint256; using Address for address; using Counters for Counters.Counter; // Equals to `bytes4(keccak256("onERC721Received(address,address,uint256,bytes)"))` // which can be also obtained as `IERC721Receiver(0).onERC721Received.selector` bytes4 private constant \_ERC721\_RECEIVED = 0x150b7a02;  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]