# 8.https://stackoverflow.com/questions/73616945/parsererror-source-project-src-contracts-interfaces-libraries-ierc165-sol-no

**T:**ParserError: Source "project:/src/contracts/Interfaces/Libraries/IERC165.sol" not found

**Q:**I have an error. I am trying to import my 'IERC165.sol' file to my 'ERC165.sol' and to my 'ERC721'. And I am receiving this error a long with two more error's that are the same just with different files.  
  
I also made sure that the file was in the folder where it should be. And now I am just confused.  
  
here is the code for the 'IERC165'  
  
// SPDX-License-Identifier: MIT pragma solidity ^0.8.0;interface IERC165 { // @notice Query if a contract implements an interface // @param interfaceID The interface identifier, as specified in ERC-165 // @dev Interface identification is specified in ERC-165. This function // uses less than 30,000 gas. // @retrun 'true' if the contract implements 'interfaceID' and // 'interfaceId' is not 0xffffffff, 'false' otherwise function supportsInterface(bytes4 interfaceID) external view returns (bool);}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
here is the code for ERC165  
  
// SPDX-License-Identifier: MIT pragma solidity ^0.8.0; import './IERC165.sol'; contract ERC165 is IERC165 { mapping (bytes4 => bool) private \_supportedInterfaces; constructor() { \_registerInterface(bytes4(keccak256('supportsInterface(bytes4)'))); } function supportsInterface(bytes4 interfaceID) external view returns (bool) { return \_supportedInterfaces[interfaceID]; } function \_registerInterface(bytes4 interfaceId) internal{ require(interfaceId != 0xfffffff, 'Invalid interface request'); \_supportedInterfaces[interfaceId] = true; } }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Here is the code for ERC721  
  
 // SPDX-License-Identifier: MIT pragma solidity ^0.8.0; import '/.ERC165.sol'; import '/.IERC721.sol'; /\*Building out the minting function a. nft to point an address b. keep track of the token ids c. keep track of token owner addresses to token ids d. keep track of how many tokens an owner address has e. create an event that emits a transfer log- contact address, where it is being minted to, the id \*/contract ERC721 is ERC165, IERC721 { // mapping solidity creates a hash table of key // pair values // Mapping from token id to the owner mapping(uint => address) private \_tokenOwner; // Mapping from owner to number of owned tokens mapping(address => uint) private \_OwnedTokensCount; // Mapping from token id to approved addresses mapping(uint256 => address) private \_tokenApprovals; /// @notice Count all NFTs assigned to an owner /// @dev NFTs assigned to zero address are considered invalid /// function throws for queries about the zero address. /// @param \_owner An address for whom to query the balance /// @return The number of NFTs owned by '\_owner', possibly zero function balanceOf(address \_owner) public override view returns(uint256) { require(\_owner != address(0), 'owner query for nonexistent token'); return \_OwnedTokensCount[\_owner]; } /// @notice Find the owner of an NFT /// @dev NFTs assigned to zero address are considered invalid, and queries /// about them do throw. /// @param \_tokenId The identifier for an NFT /// @return The address of the owner of the NFT function ownerOf(uint256 \_tokenId) public view override returns (address) { address owner = \_tokenOwner[\_tokenId]; require(owner != address(0), 'owner query for nonexistent token'); return owner; } function \_exists(uint256 tokenId) internal view returns (bool){ // setting the address of nft owner to check the mapping // of the addres from the token owner at the tokenId address owner = \_tokenOwner[tokenId]; // return truthiness that address in not zero return owner != address (0);}function \_mint(address to, uint256 tokenId) internal virtual { // requires the address isn't zero require(to != address(0), 'ERC721: minting to the zero address'); // requires that he token does not already exist require(!\_exists(tokenId), 'ERC721: token already minted'); // we are adding a new address with a token id for minting \_tokenOwner[tokenId] = to; // keeping track of each address that is minting and adding one to the count \_OwnedTokensCount[to] += 1; emit Transfer(address (0), to, tokenId); } // @notice Transfer owner ship of an NFT -- THE REAL CALLER IS RESPOSIBLE // TO CONFIRM THAT '\_to' IS CAPABLE OF RECEIVING NFTS OR ELSE // THEY MAY BE PERMANENTLY LOST // @dev Throws untess 'msg.sender' is the current owner, an authorized // operator, or the approved address for this NFT. Throws if '\_from' is // not current owner, Throws if '\_to' is the zero address. Throws if // '\_tokenId' is not a valid NFT. // @param \_from The current owner of the NFT // @param \_to the new owner // @param \_tokenId The NFT to transferfunction \_transferFrom(address \_from, address \_to, uint256 \_tokenId) internal { require(\_to != address(0), 'Error - ERC721 Tr.ansfer to the zero address'); require(ownerOf(\_tokenId) == \_from, 'Trying to transfer a token address does not own'); \_OwnedTokensCount[\_from] -= 1; \_OwnedTokensCount[\_to] += 1; \_tokenOwner[\_tokenId] = \_to; emit Transfer(\_from, \_to, \_tokenId); } function transferFrom(address \_from, address \_to, uint256 \_tokenId) override public { \_transferFrom(\_from, \_to, \_tokenId); }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

1 **Answer**

**A1:**Typo error in the import of interfaces..  
  
 import './ERC165.sol'; import './IERC721.sol';   
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
instead of  
  
 import '/.ERC165.sol'; import '/.IERC721.sol';   
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

**C1:**i've already fixed that but, I have another one. In the ERC165...