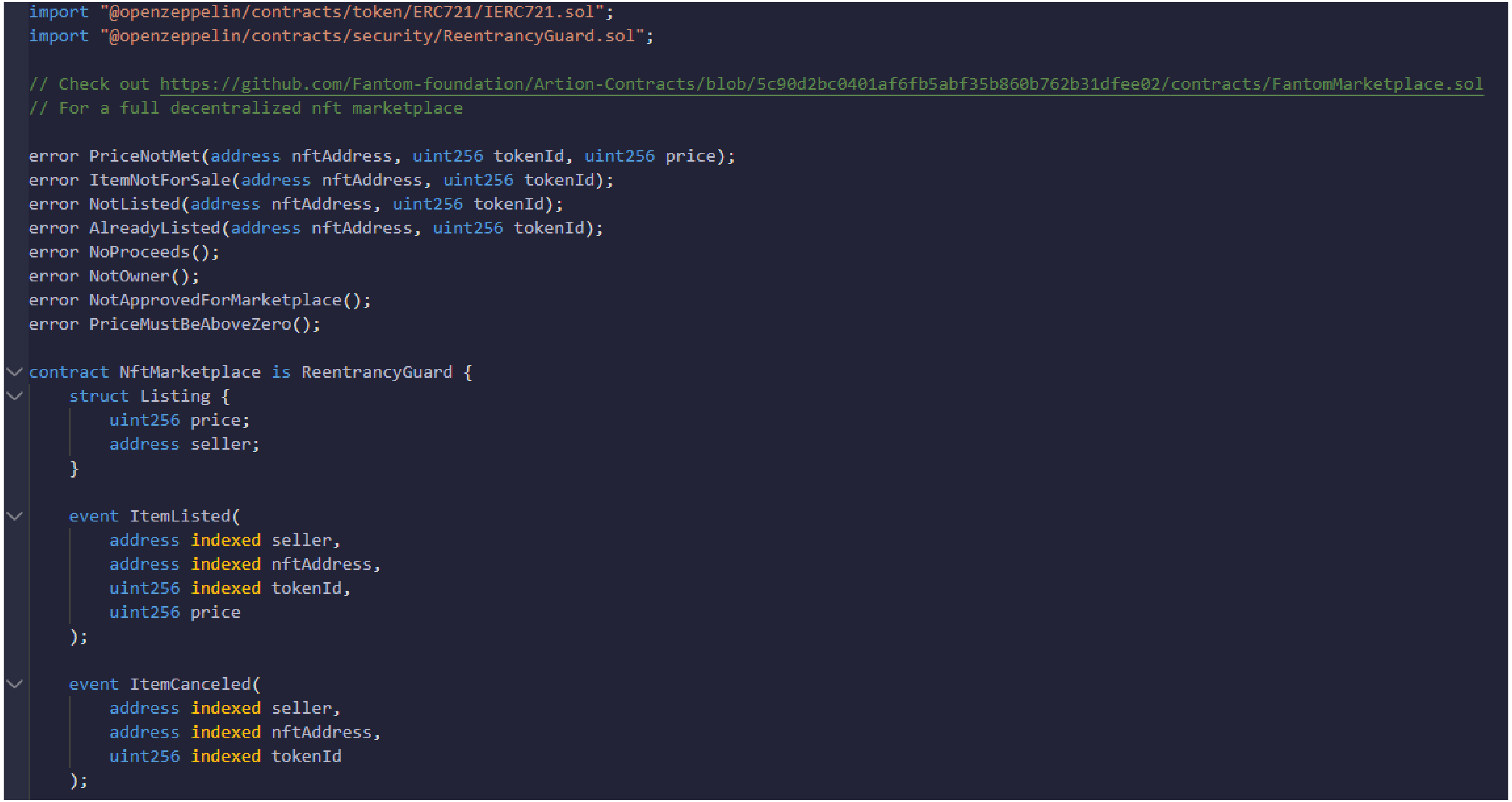
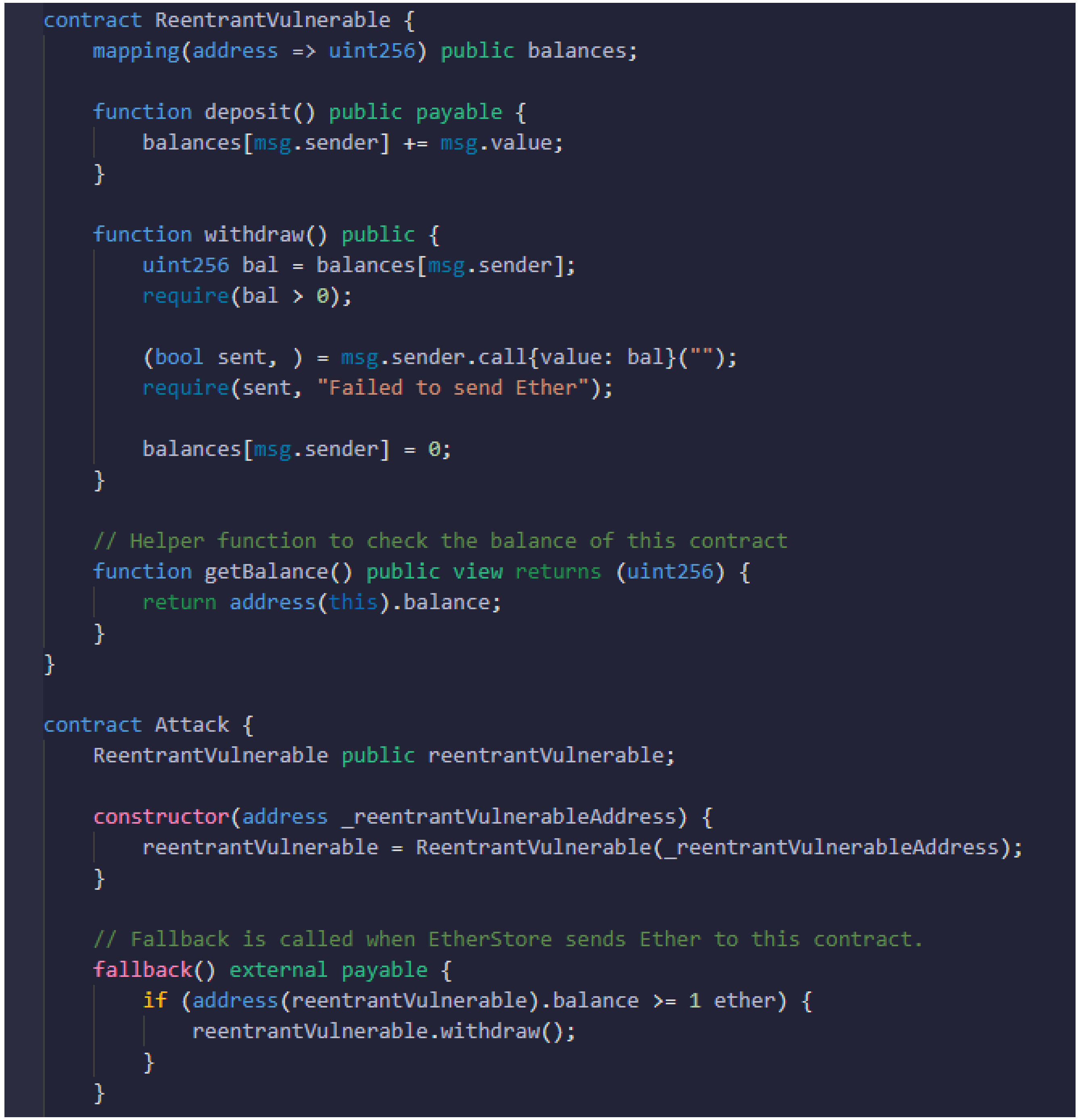
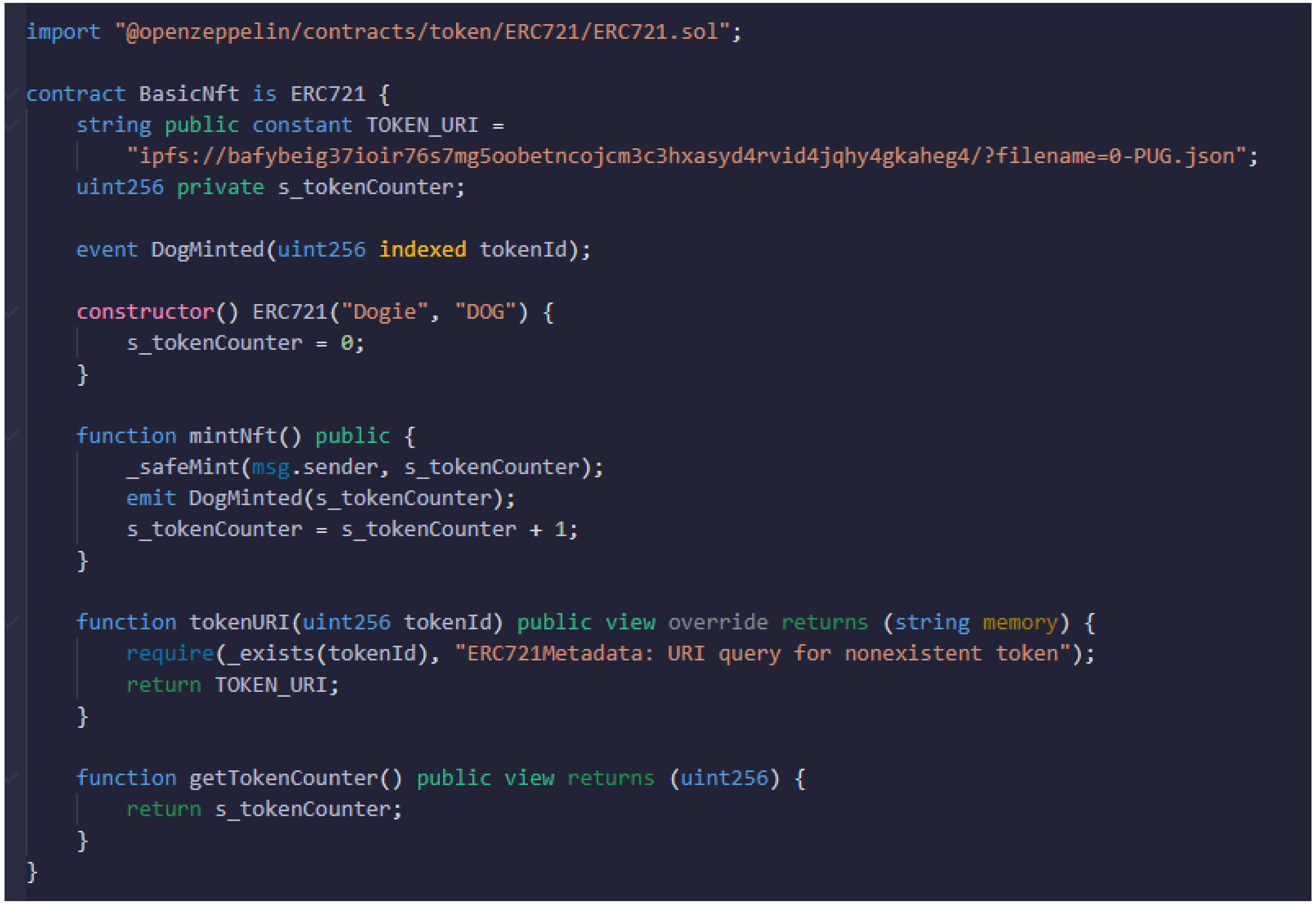
NextJS NFT Marketplace



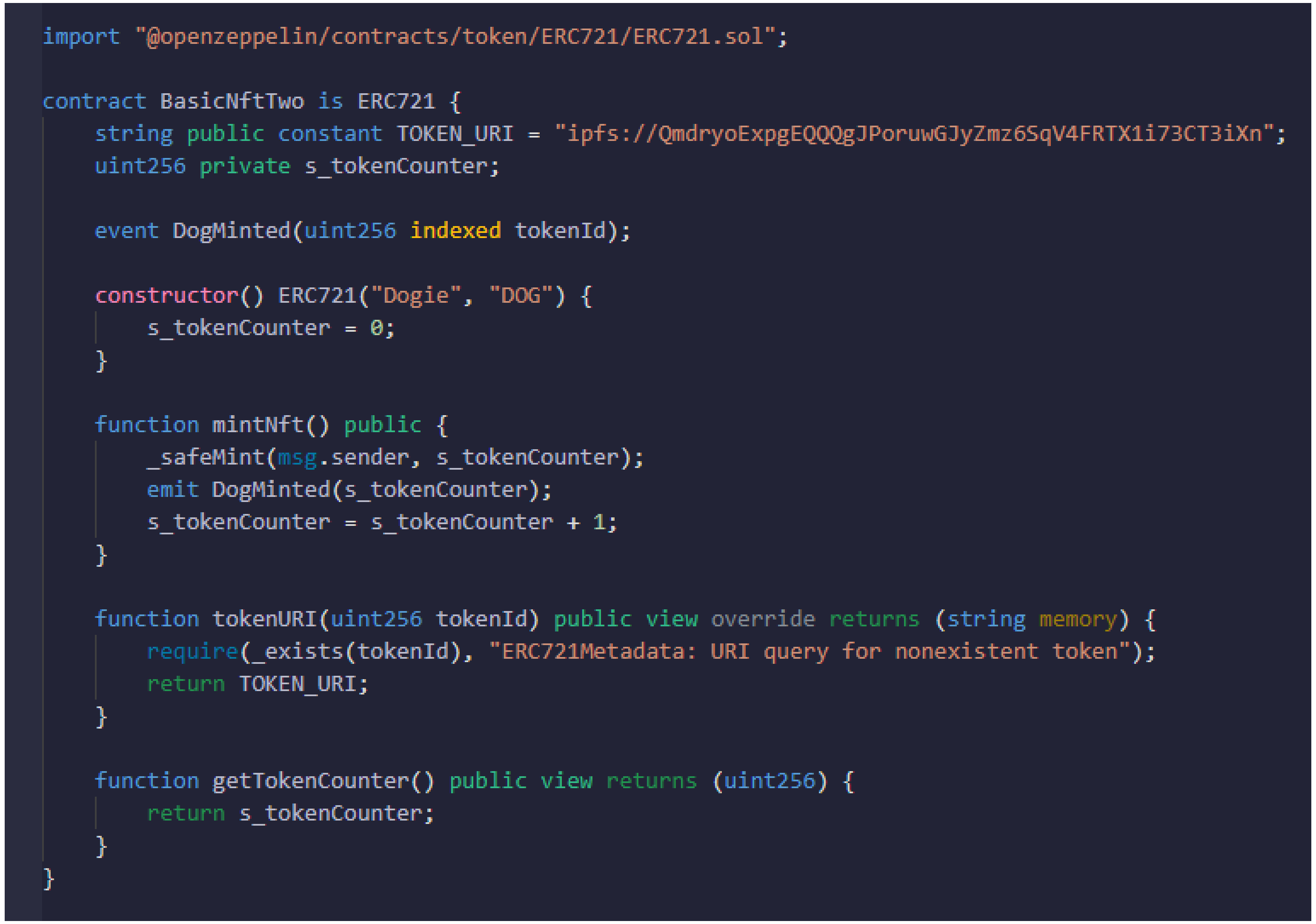
```
import "@openzeppelin/contracts/token/ERC721/IERC721.sol";
import "@openzeppelin/contracts/security/ReentrancyGuard.sol";

// Check out https://github.com/Fantom-foundation/Artion-Contracts/blob/5c90d2bc0401af6fb5abf35b860b762b31dfee02/contracts/FantomMarketplace.sol
// For a full decentralized nft marketplace

error PriceNotMet(address nftAddress, uint256 tokenId, uint256 price);
error ItemNotForSale(address nftAddress, uint256 tokenId);
error NotListed(address nftAddress, uint256 tokenId);
error AlreadyListed(address nftAddress, uint256 tokenId);
error NoProceeds();
error NotOwner();
error NotApprovedForMarketplace();
error PriceMustBeAboveZero();

contract NftMarketplace is ReentrancyGuard {
    struct Listing {
        uint256 price;
        address seller;
    }

    event ItemListed(
        address indexed seller,
        address indexed nftAddress,
        uint256 indexed tokenId,
        uint256 price
    );

    event ItemCanceled(
        address indexed seller,
        address indexed nftAddress,
        uint256 indexed tokenId
    );
```



```solidity
contract ReentrantVulnerable {
    mapping(address => uint256) public balances;

    function deposit() public payable {
        balances[msg.sender] += msg.value;
    }

    function withdraw() public {
        uint256 bal = balances[msg.sender];
        require(bal > 0);

        (bool sent, ) = msg.sender.call{value: bal}("");
        require(sent, "Failed to send Ether");

        balances[msg.sender] = 0;
    }

    // Helper function to check the balance of this contract
    function getBalance() public view returns (uint256) {
        return address(this).balance;
    }
}

contract Attack {
    ReentrantVulnerable public reentrantVulnerable;

    constructor(address _reentrantVulnerableAddress) {
        reentrantVulnerable = ReentrantVulnerable(_reentrantVulnerableAddress);
    }

    // Fallback is called when EtherStore sends Ether to this contract.
    fallback() external payable {
        if (address(reentrantVulnerable).balance >= 1 ether) {
            reentrantVulnerable.withdraw();
        }
    }
```



```solidity
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";

contract BasicNft is ERC721 {
    string public constant TOKEN_URI =
        "ipfs://bafybeig37ioir76s7mg5oobetncojcm3c3hxasyd4rvid4jqhy4gkaheg4/?filename=0-PUG.json";
    uint256 private s_tokenCounter;

    event DogMinted(uint256 indexed tokenId);

    constructor() ERC721("Dogie", "DOG") {
        s_tokenCounter = 0;
    }

    function mintNft() public {
        _safeMint(msg.sender, s_tokenCounter);
        emit DogMinted(s_tokenCounter);
        s_tokenCounter = s_tokenCounter + 1;
    }

    function tokenURI(uint256 tokenId) public view override returns (string memory) {
        require(_exists(tokenId), "ERC721Metadata: URI query for nonexistent token");
        return TOKEN_URI;
    }

    function getTokenCounter() public view returns (uint256) {
        return s_tokenCounter;
    }
}
```



```solidity
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";

contract BasicNftTwo is ERC721 {
    string public constant TOKEN_URI = "ipfs://QmdryoExpgEQQQgJPoruwGJyZmz6SqV4FRTX1i73CT3iXn";
    uint256 private s_tokenCounter;

    event DogMinted(uint256 indexed tokenId);

    constructor() ERC721("Dogie", "DOG") {
        s_tokenCounter = 0;
    }

    function mintNft() public {
        _safeMint(msg.sender, s_tokenCounter);
        emit DogMinted(s_tokenCounter);
        s_tokenCounter = s_tokenCounter + 1;
    }

    function tokenURI(uint256 tokenId) public view override returns (string memory) {
        require(_exists(tokenId), "ERC721Metadata: URI query for nonexistent token");
        return TOKEN_URI;
    }

    function getTokenCounter() public view returns (uint256) {
        return s_tokenCounter;
    }
}
```