OtherWorlds Contract Guide

The contract imports various contracts from the OpenZeppelin library, such as ERC721, ERC721Enumerable, Pausable, Ownable, and Counters.

The contract definition begins with contract OtherWorlds is ERC721, ERC721Enumerable, Pausable, Ownable. This means that the OtherWorlds contract inherits from these imported contracts, inheriting their functionalities and state variables.

Inside the contract, it defines several property variables using Counters.Counter to create a counter for token IDs. It also sets the contract owner, mint price, and maximum supply of tokens.

The contract constructor constructor() ERC721("OtherWorlds", "HEART") initializes the contract. It increments the token ID counter.

The Witchdraw function allows the contract owner to withdraw the balance of the contract if it is greater than zero.

The transfer function enables transferring funds from the contract to a specified address.

The sendNFT function allows sending an NFT (non-fungible token) from the caller's address to a specified address.

The buyNFT function allows users to buy an NFT by transferring the specified amount of ether to the current owner of the NFT. It performs several validations and transfers ownership and funds accordingly.

The safeMint function enables the creation of new NFTs by minting them to a specified address. It performs validations on the amount of ether sent and the maximum supply of tokens.

The createNFT function allows the contract owner to create an NFT by minting it to their address. It performs validations on the amount of ether sent and increments the maximum supply of tokens.

The pause and unpause functions allow the contract owner to pause and unpause the contract, respectively.

The _beforeTokenTransfer function is an internal function used as a hook to perform actions before transferring tokens between addresses. It ensures that the contract is not paused before performing the transfer.

The _baseURI function returns the base URI for token metadata.

The supportsInterface function overrides the corresponding function from the inherited contracts and ensures that the contract supports the specified interface.