title: ERC-721 Non-Fungible Token Standard description: lang: en

Introduction {#introduction}

What is a Non-Fungible Token?

A Non-Fungible Token (NFT) is used to identify something or someone in a unique way. This type of Token is perfect to be used on platforms that offer collectible items, access keys, lottery tickets, numbered seats for concerts and sports matches, etc. This special type of Token has amazing possibilities so it deserves a proper Standard, the ERC-721 came to solve that!

What is ERC-721?

The ERC-721 introduces a standard for NFT, in other words, this type of Token is unique and can have different value than another Token from the same Smart Contract, maybe due to its age, rarity or even something else like its visual. Wait, visual?

Yes! All NFTs have a `uint256` variable called `tokenId`, so for any ERC-721 Contract, the pair `contract address, uint256 tokenId` must be globally unique. That said, a dapp can have a "converter" that uses the `tokenId` as input and outputs an image of something cool, like zombies, weapons, skills or amazing kitties!

Prerequisites {#prerequisites}

- Accounts
- Smart Contracts
- Token standards

Body {#body}

The ERC-721 (Ethereum Request for Comments 721), proposed by William Entriken, Dieter Shirley, Jacob Evans, Nastassia Sachs in January 2018, is a Non-Fungible Token Standard that implements an API for tokens within Smart Contracts.

It provides functionalities like to transfer tokens from one account to another, to get the current token balance of an account, to get the owner of a specific token and also the total supply of the token available on the network. Besides these it also has some other functionalities like to approve that an amount of token from an account can be moved by a third party account.

If a Smart Contract implements the following methods and events it can be called an ERC-721 Non-Fungible Token Contract and, once deployed, it will be responsible to keep track of the created tokens on Ethereum.

From EIP-721:

Methods {#methods}

```
solidity function balanceOf(address _owner) external view returns (uint256); function ownerOf(uint256 _tokenId) external view returns (address); function safeTransferFrom(address
_from, address _to, uint256 _tokenId, bytes data) external payable; function safeTransferFrom(address _from, address _to, uint256 _tokenId) external payable; function
transferFrom(address _from, address _to, uint256 _tokenId) external payable; function approve(address _approved, uint256 _tokenId) external payable; function
setApprovalForAll(address _operator, bool _approved) external; function getApproved(uint256 _tokenId) external view returns (address); function isApprovedForAll(address _owner,
address _operator) external view returns (bool);
```

Events {#events}

```
solidity event Transfer(address indexed _from, address indexed _to, uint256 indexed _tokenId); event Approval(address indexed _owner, address indexed _approved, uint256 indexed
_tokenId); event ApprovalForAll(address indexed _owner, address indexed _operator, bool _approved);
```

Examples {#web3py-example}

Let's see how a Standard is so important to make things simple for us to inspect any ERC-721 Token Contract on Ethereum. We just need the Contract Application Binary Interface (ABI) to create an interface to any ERC-721 Token. As you can see below we will use a simplified ABI, to make it a low friction example.

Web3.py Example {#web3py-example}

First, make sure you have installed Web3.py Python library:

```
pip install web3
```

```python from web3 import Web3 from web3._utils.events import get_event_data

w3 = Web3(Web3.HTTPProvider("https://cloudflare-eth.com"))

ck_token_addr = "0x06012c8cf97BEaD5deAe237070F9587f8E7A266d" # CryptoKitties Contract

acc_address = "0xb1690C08E213a35Ed9bAb7B318DE14420FB57d8C" # CryptoKitties Sales Auction

# This is a simplified Contract Application Binary Interface (ABI) of an ERC-721 NFT Contract.

# It will expose only the methods: balanceOf(address), name(), ownerOf(tokenId), symbol(), totalSupply()

simplified_abi = [ { 'inputs': [{'internalType': 'address', 'name': 'owner', 'type': 'address'}], 'name': 'balanceOf', 'outputs': [{'internalType': 'uint256', 'name': '', 'type': 'uint256'}], 'payable': False, 'stateMutability': 'view', 'type': 'function', 'constant': True }, { 'inputs': [], 'name': 'name', 'outputs': [{'internalType': 'string', 'name': '', 'type': 'string'}], 'stateMutability': 'view', 'type': 'function', 'constant': True }, { 'inputs': [{'internalType': 'uint256', 'name': 'tokenId', 'type': 'uint256'}], 'name': 'ownerOf', 'outputs': [{'internalType': 'address', 'name': '', 'type': 'address'}], 'payable': False, 'stateMutability': 'view', 'type': 'function', 'constant': True }, { 'inputs': [], 'name': 'symbol', 'outputs': [{'internalType': 'string', 'name': '', 'type': 'string'}], 'stateMutability': 'view', 'type': 'function', 'constant': True }, { 'inputs': [], 'name': 'totalSupply', 'outputs': [{'internalType': 'uint256', 'name': '', 'type': 'uint256'}], 'stateMutability': 'view', 'type': 'function', 'constant': True }, ]

ck_extra_abi = [ { 'inputs': [], 'name': 'pregnantKitties', 'outputs': [{'name': '', 'type': 'uint256'}], 'payable': False, 'stateMutability': 'view', 'type': 'function', 'constant': True }, { 'inputs': [{'name': '_kittyId', 'type': 'uint256'}], 'name': 'isPregnant', 'outputs': [{'name': '', 'type': 'bool'}], 'payable': False, 'stateMutability': 'view', 'type': 'function', 'constant': True } ]

ck_contract = w3.eth.contract(address=w3.to_checksum_address(ck_token_addr), abi=simplified_abi+ck_extra_abi) name = ck_contract.functions.name().call() symbol = ck_contract.functions.symbol().call() kitties_auctions = ck_contract.functions.balanceOf(acc_address).call() print(f"{name} [{symbol}] NFTs in Auctions: {kitties_auctions}")

pregnant_kitties = ck_contract.functions.pregnantKitties().call() print(f"{name} [{symbol}] NFTs Pregnants: {pregnant_kitties}")

# Using the Transfer Event ABI to get info about transferred Kitties.

tx_event_abi = { 'anonymous': False, 'inputs': [ {'indexed': False, 'name': 'from', 'type': 'address'}, {'indexed': False, 'name': 'to', 'type': 'address'}, {'indexed': False, 'name': 'tokenId', 'type': 'uint256'}], 'name': 'Transfer', 'type': 'event' }
```

We need the event's signature to filter the logs

event_signature = w3.keccak(text="Transfer(address,address,uint256)").hex()

logs = w3.eth.get_logs({ "fromBlock": w3.eth.block_number - 120, "address": w3.to_checksum_address(ck_token_addr), "topics": [event_signature] })

Notes:

- Increase the number of blocks up from 120 if no Transfer event is returned.

- If you didn't find any Transfer event you can also try to get a tokenId at:

https://etherscan.io/address/0x06012c8cf97BEaD5deAe237070F9587f8E7A266d#events

Click to expand the event's logs and copy its "tokenId" argument

recent_tx = [get_event_data(w3.codec, tx_event_abi, log)["args"] for log in logs]

if recent_tx: kitty_id = recent_tx[0]['tokenId'] # Paste the "tokenId" here from the link above is_pregnant = ck_contract.functions.isPregnant(kitty_id).call() print(f"{name} [{symbol}] NFTs {kitty_id} is pregnant: {is_pregnant}") ```

CryptoKitties Contract has some interesting Events other than the Standard ones.

Let's check two of them, `Pregnant` and `Birth`.

```python

## Using the Pregnant and Birth Events ABI to get info about new Kitties.

ck_extra_events_abi = [ { 'anonymous': False, 'inputs': [ {'indexed': False, 'name': 'owner', 'type': 'address'}, {'indexed': False, 'name': 'matronId', 'type': 'uint256'}, {'indexed': False, 'name': 'sireId', 'type': 'uint256'}, {'indexed': False, 'name': 'cooldownEndBlock', 'type': 'uint256'}], 'name': 'Pregnant', 'type': 'event' }, { 'anonymous': False, 'inputs': [ {'indexed': False, 'name': 'owner', 'type': 'address'}, {'indexed': False, 'name': 'kittyId', 'type': 'uint256'}, {'indexed': False, 'name': 'matronId', 'type': 'uint256'}, {'indexed': False, 'name': 'sireId', 'type': 'uint256'}, {'indexed': False, 'name': 'genes', 'type': 'uint256'}], 'name': 'Birth', 'type': 'event' }]

## We need the event's signature to filter the logs

ck_event_signatures = [ w3.keccak(text="Pregnant(address,uint256,uint256,uint256)").hex(), w3.keccak(text="Birth(address,uint256,uint256,uint256,uint256)").hex(), ]

## Here is a Pregnant Event:

## - https://etherscan.io/tx/0xc97eb514a41004acc447ac9d0d6a27ea6da305ac8b877dff37e49db42e1

pregnant_logs = w3.eth.get_logs({ "fromBlock": w3.eth.block_number - 120, "address": w3.to_checksum_address(ck_token_addr), "topics": [ck_event_signatures[0]] })

recent_pregnants = [get_event_data(w3.codec, ck_extra_events_abi[0], log)["args"] for log in pregnant_logs]

## Here is a Birth Event:

## - https://etherscan.io/tx/0x3978028e08a25bb4c44f7877eb3573b9644309c044bf087e335397f16356

birth_logs = w3.eth.get_logs({ "fromBlock": w3.eth.block_number - 120, "address": w3.to_checksum_address(ck_token_addr), "topics": [ck_event_signatures[1]] })

recent_births = [get_event_data(w3.codec, ck_extra_events_abi[1], log)["args"] for log in birth_logs] ```

### Popular NFTs {#popular-nfts}

- Etherscan NFT Tracker list the top NFT on Ethereum by transfers volume.
- CryptoKitties is a game centered around breedable, collectible, and oh-so-adorable creatures we call CryptoKitties.
- Sorare is a global fantasy football game where you can collect limited editions collectibles, manage your teams and compete to earn prizes.
- The Ethereum Name Service (ENS) offers a secure & decentralized way to address resources both on and off the blockchain using simple, human-readable names.
- POAP delivers free NFTs to people who attend events or complete specific actions. POAPs are free to create and distribute.
- Unstoppable Domains is a San Francisco-based company building domains on blockchains. Blockchain domains replace cryptocurrency addresses with human-readable names and can be used to enable censorship-resistant websites.
- Gods Unchained Cards is a TCG on the Ethereum blockchain that uses NFT's to bring real ownership to in-game assets.
- Bored Ape Yacht Club is a collection of 10,000 unique NFTs, which, as well as being a provably-rare piece of art, acts as a membership token to the club, providing member perks and benefits that increase over time as a result of community efforts.

### Further reading {#further-reading}

- EIP-721: ERC-721 Non-Fungible Token Standard
- OpenZeppelin - ERC-721 Docs
- OpenZeppelin - ERC-721 Implementation
- Alchemy NFT API
```