

Information Technologies for Industrial Engineers

เทคโนโลยีสารสนเทศสำหรับวิศวกรอุตสาหกรรม

Smart contract applications

Decentralized lottery

Decentralized lottery

- Code
- Unit converter

ERC Token

What is a token?

- Something of value
 - Currency
 - Voting right
 - Stock
- Token standard
 - EIP (*Ethereum Improvement Proposal*)
 - Guideline
 - ERC (*Ethereum Request for Comments*)
 - Implementation

Popular token standards

- **ERC-20**
 - Fungible tokens
 - Most used for representing currency
- **ERC-721**
 - Non-fungible tokens (NFTs)
 - Most used for representing digital artwork and collectibles
- **ERC-1155**
 - Multi-token standard
 - Combining the abilities of ERC-20 and ERC-720

Timeline

Source

Token list

- <https://etherscan.io/tokens>

Let's make your own ERC-20 token.

- [Contract generator](#)
- [Source code](#)

IPFS

Interplanetary File System

Decentralized data storage

- DApps need to store/retrieve data in a decentralized manner.
- Saving data in the blockchain is very expensive.
- We need to save data **off-chain**.

IPFS

- *Interplanetary File System*
 - Decentralized off-chain solution for data storage.
- When we put data on IPFS, we obtain **content identifier (CID)** that uniquely identifies the data.
 - For example, `QmTHUf5DiynRc5WRJBMcZYtigMWqkdtJVxuQVKoAjMPnoB`
 - We store CID on the ethereum block chain.
- Note: `Swarm` is an alternative to IPFS.

IPFS

- Aims to replace HTTP.
- Peer-to-peer (based on bit-torrent)
 - Network consists of multiple *nodes*.
- Data is permanent and cannot be deleted or modified
 - Data are already distributed to other nodes.

Content/Location addressing

- **Location addressing**

- `https://cmu.ac.th/file.pdf`
- Who ever controls that location control the content.

- **Content addressing**

- Files are based on **where they are**, but on **what they are**.
- There is no location of the files.
- No one controls the files.

Pinning

- Mechanism that allows IPFS to always keep a given content and never remove it.

Pinata

- IPFS pinning service.
- <https://www.pinata.cloud/>

Gateway

- Allows browsers to access IPFS.
- **Local gateway** (need to run IPFS client)
 - `http://localhost:8080/ipfs/[CID]` (ex)
- **Private gateway**
- **Public gateway**
 - `https://ipfs.io/ipfs/[CID]` (ex)
 - `https://gateway.pinata.cloud/ipfs/[CID]` (ex)

Non-Fungible Token (NFT)

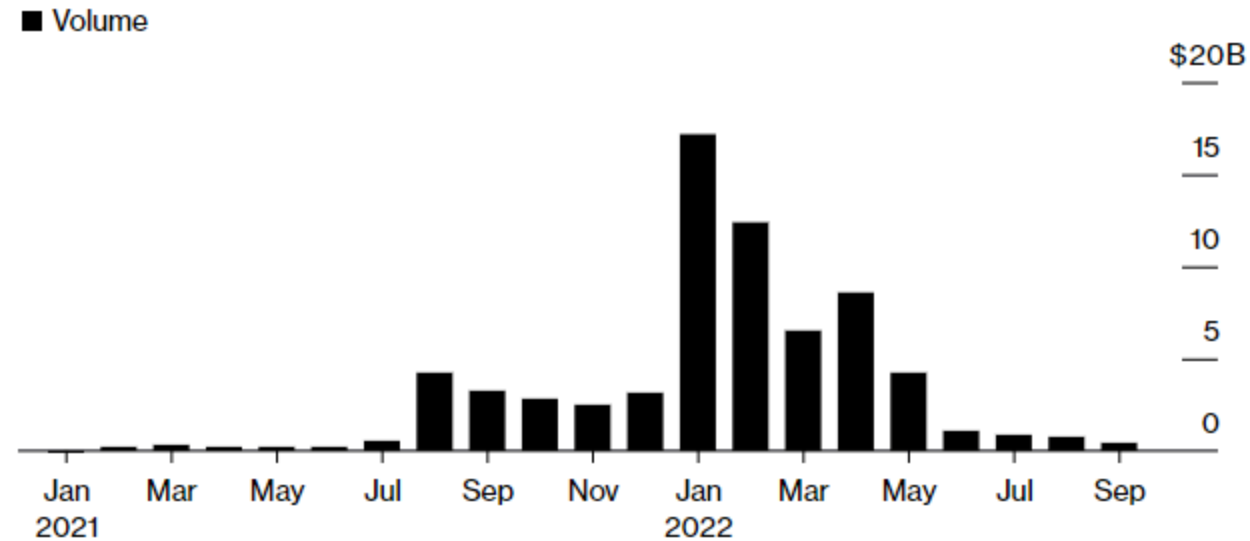
Non-Fungible Token (NFT)?

- Cryptographic assets on a blockchain with unique identification codes and metadata that distinguish them from each other.
- Cannot be traded or exchanged at equivalency.
- NFTs can represent real-world items like artwork and real estate.

Hype?

Volume Drop

NFT monthly volume has dropped 97% from 2022 highs



Source: Dune Analytics; Dashboard by @hildobby

Note: Cumulative data from OpenSea, NFTX, LarvaLabs, LooksRare, SuperRare, Rarible, Foundation

Standard

- ERC-721 standard
 - Most common
- The ERC-1155 standard takes the concept further.
 - Reducing the transaction and storage costs required for NFTs.
 - Matching multiple types of non-fungible tokens into a single contract.

NFTs in action

- CLOAKS #19345
 - Contract
 - Token URI
 - Image

Create your own NFTs (simple)

OpenSea Testnet

Implementing ERC721

- OpenZeppelin

The screenshot displays the OpenZeppelin contract generator interface for ERC721. The 'ERC721' tab is selected in the top navigation bar. The 'SETTINGS' section on the left shows 'Name' as 'MyToken' and 'Symbol' as 'MTK'. The 'Base URI' is set to 'https://...'. Under the 'FEATURES' section, 'Mintable', 'Auto Increment Ids', and 'URI Storage' are checked. In the 'ACCESS CONTROL' section, 'Ownable' is selected. Three red arrows highlight key elements: the first points to the 'ERC721' tab, the second points to the 'URI Storage' feature, and the third points to the generated Solidity code on the right. The code includes imports for OpenZeppelin contracts, a constructor for 'MyToken' and 'MTK', a 'safeMint' function, and overrides for '_burn' and 'tokenURI'.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.9;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Counters.sol";

contract MyToken is ERC721, ERC721URIStorage, Ownable {
    using Counters for Counters.Counter;

    Counters.Counter private _tokenIdCounter;

    constructor() ERC721("MyToken", "MTK") {}

    function safeMint(address to, string memory uri) public onlyOwner {
        uint256 tokenId = _tokenIdCounter.current();
        _tokenIdCounter.increment();
        _safeMint(to, tokenId);
        _setTokenURI(tokenId, uri);
    }

    // The following functions are overrides required by Solidity.

    function _burn(uint256 tokenId) internal override(ERC721, ERC721URIStorage) {
        super._burn(tokenId);
    }

    function tokenURI(uint256 tokenId)
        public
        view
        override(ERC721, ERC721URIStorage)
        returns (string memory)
    {}
}
```

Create your own NFTs (real)

1. Create image

2. Upload to IPFS

2.1 Get `image_url`

3. Create metadata

3.1 Save as `JSON` file

4. Upload metadata file to IPFS

4.1 Get `token_uri`

5. Deploy contract

6. Mint token

Metadata

```
{
  "name": "<<name>>",
  "description": "<<description>>",
  "external_url": "<<external_link>>",
  "image": "<<image_url>>",
  "attributes": [
    {
      "trait_type": "Personality",
      "value": "Sad"
    },
    {
      "trait_type": "Level",
      "value": 5
    },
    {
      "display_type": "boost_number",
      "trait_type": "Power",
      "value": 40
    }
  ]
}
```